Iulian ACIOBĂNIȚEI (coord.) Mihai TOGAN (coord.)

INFORMATICĂ

Culegere de probleme de tip grilă
pentru admiterea în învățământul superior

ISBN 978-973-640-355-2

Editura Academiei Tehnice Militare "Ferdinand I"
București, 2025

Autorii, cadre didactice ale Facultății de Calculatoare și Securitate Cibernetică din instituție, au contribuit în mod egal la elaborarea prezentei culegeri, prin compunerea de probleme originale.

Iulian ACIOBĂNIȚEI
Dan AVRAM
Emil BUREACĂ
Alexandra BUZĂȚOIU
Andrei BRÎNZEA
Mihai COCA
Luca CORĂȚU
George HARIGA
Dragoș IOANA
Mirabela MEDVEI
Ștefania NIȚĂ
Ștefania ȘTEFĂNESCU
Iulian TIȚĂ
Mihai TOGAN
Ștefan TOMA
Adina VAMAN

PREFAȚĂ

Această lucrare se dorește a fi un ghid valoros pentru toți cei care se pregătesc pentru examenul de admitere în Academia Tehnică Militară „Ferdinand I”, precum și pentru liceenii care vor susține un examen de tip test grilă în domeniul informaticii.

Autorii au structurat culegerea astfel încât să fie accesibilă atât celor care doresc să învețe noțiuni de bază, cât și celor care vor să abordeze probleme mai complexe. Conținutul lucrării este în concordanță atât cu programa analitică stabilită pentru examenul de bacalaureat (profil matematică-informatică) cât și cu cea prevăzută pentru examenul de admitere în Academia Tehnică Militară “Ferdinand I”.

Prin concentrarea pe concepte esențiale precum tipurile de date de bază, algoritmică, vectori și matrice, grafuri neorientate, arbori și tehnici avansate precum backtracking sau analiza complexității algoritmilor, lucrarea asigură o pregătire completă și diversificată.

Pe parcursul lucrării, autorii au ținut cont de necesitatea de a oferi nu doar răspunsuri corecte, ci și explicații clare pentru fiecare problemă, ceea ce va ajuta la aprofundarea cunoștințelor și înțelegerea logicii din spatele rezolvării acestora.

Formatul problemelor este cel folosit la exemenul de admitere în Academia Tehnică Militară “Ferdinand I”. Fiecare problemă are 4 variante de răspuns, dintre care doar unul este corect. În cadrul formulărilor de probleme, pentru a elimina detaliile redundante, nu sunt menționate bibliotecile incluse sau diverse declarări și inițializări.

Chiar dacă sunt deja disponibile public pe site-ul instituției, pentru ușurința candidaților, chestionarele de admitere utilizate în perioada 2021-2024 în cadrul examenelor de admitere la licență din Academia Tehnică Militară "Ferdinand I." au fost și ele agregate în ultimul capitol.

Autorii au convingerea că această culegere de probleme vine în sprijinul candidaților la admiterea în Academia Tehnică Militară "Ferdinand I", facilitându-le pregătirea pentru examen.

Vă dorim mult SUCCES!

Aducem mulțumiri studenților care au ajutat la elaborarea acestei culegeri:

Claudiu ACOSTI
Marian AIONESEI
Dimitrie-Adrian ALBU
Valentin AXINESCU
Patricia BARBUȚĂ
Alin BUGA
Ionuț BURCEA
Bogdan CĂPRIȚĂ
Raluca CARAN
Andrei-Daniel CODREANU
Claudia-Ioana DASCĂLESCU
Dariana GRAURE
Florentin-Marian IVAN
Tudor LEPĂDATU
Bogdan MECLEA
Sorin-Ionuț MIHALI
Diana NEGUȚ
Daria PASCU
Alexandru-Sebastian PILĂ
George-Mihai ROȘCA
Bianca SÎRBU
Doru-Dumitru TĂNASE

CUPRINS

PROBLEME DE NIVEL UȘOR-MEDIU. ENUNȚURI
INSTRUCȚIUNI ȘI OPERATORI

1. Menționați ce afișează următorul program.

```
void main(){
      int a=5,  b=4,  c=3;
      if (a>b && b<c)
            if (a<c)
                  printf("1");
            else
                  printf("2");
      else
            if (b  >  c)
                  printf("3");
            else
                  printf("4");
}
```

a) 3 **b)** 1 **c)** 2 **d)** 4

2. Se consideră programul de mai jos. Ce valoarea se va afișa pe ecran?

```
void main() {
      int x=0,y=10,z=4;
      x=y/z+2;
      printf("%d", x);
}
```

a) 2 **b)** 4 **c)** 5 **d)** 4.2

3. Precizați ce valoare are variabila *a* după execuția secvenței următoare de instrucțiuni:

```
int a = 5;
int b = 7;
a++;
a += --b;
```

a) 13 **b)** 11 **c)** 10 **d)** 12

4. Se consideră subprogramul de mai jos. Ce afișează acesta?

```
void main(){
      int a = -2;
      printf("%d %d", !a, !!a);
}
```

a) 0 1 **b)** 1 0 **c)** 0 -2 **d)** 1 -2

5. Se consideră subprogramul de mai jos. Menționați ce va afișa apelul funcției *cifre* pentru *n = 12345*.

```
void cifre(int n) {
      while (n > 0) {
            printf("%d ", n % 10);
            n /= 10;
      }
}
```

a) 5 4 3 2 1 **b)** 5 4 3 2 1 0 **c)** 0 1 2 3 4 5 **d)** 1 2 3 4 5

6. Care este valoarea constantei *a* pentru care următoarea secvență de program va afișa pe ecran șirul *Admitere_ATM*?

```
int x = 2025;
while (x > a)
{
      x--;
      if (x % 4)   printf("ATM");
      else  printf("Admitere_");
}
```

a) 2021 **b)** 2022 **c)** 2023 **d)** 2024

7. Menționați ce afișează programul de mai jos, considerând că sunt citite numere întregi pozitive.

```
void main() {
      int i, d, n = 10, m=-1 ;
      for (i = 0; i < n; i++)
      {
            scanf("%d", &d);
            m = (m < d) ? d : m;
      }
      printf("%d", m);
}
```

a) minimul elementelor introduse
b) maximul elementelor introduse
c) primul element introdus
d) ultimul element introdus

8. Precizați care dintre scenariile următoare fac ca rezultatul evaluării următoarei expresii să fie 1:

```
(a < b) == (c < d)
```

a) *a < b și c > d*
b) *a > b și c < d*
c) *a >=b și c < d*
d) *a >=b și c >=d*

9. Menționați efectul execuției următorului program.

```
void main(){
	int i = 10;
	int cnt = 0;
	while (--i)
	{
		if (i % 2)
			i--;
		cnt++;
	}
	printf("%d\n", cnt);
}
```

a) Programul generează o buclă infinită
b) Programul afișează valoarea 5
c) Programul afișează valoarea 4
d) Programul afișează valoarea 6

10. Precizați ce valori au variabilele *x* și *y*, după execuția secvenței următoare de cod:

```
int x = 2025, y;
y = ++x + x / 10 + x % 10;
```

a) x = 2026, y = 2032
b) x = 2025, y = 2032
c) x = 2026, y = 2234
d) x = 2026, y = 2234.6

11. Când va returna subprogramul *e* valoarea *1*?

```
int e(int x) {
      if (x < 2) return 0;
      if (x == 2) return 1;
      if (!(x % 2)) return 0;
      for (int j = 3; j * j <= x; j += 2)
            if (!(x % j)) return 0;
return 1;}
```

a) Atunci când *x* este pătrat perfect
b) Atunci când *x* este un număr impar mai mare decât *2*
c) Atunci când *x* este divizibil cu *2*
d) Atunci când *x* este un număr prim

12. Se consideră subprogramul de mai jos, unde *x* și *n* sunt numere întregi. Menționați ce va afișa apelul funcției *gaseste* pentru *n = 100*.

```
int verifica(int x) {
      if (x < 2)
            return 0;
      for (int i = 2; i <= sqrt(x); i++)
            if (x % i == 0)
                  return 0;
      return 1;
}
int gaseste(int n) {
      for (int i = n - 1; i >= 2; i--)
            if (verifica(i))
                  return i;
      return -1;
}
```

a) 97 **b)** 1 **c)** 100 **d)** 0

13. Se consideră funcția de mai jos. Menționați ce calculează aceasta considerând *a* și *b* două numere naturale și ce valoare va returna apelul *f(48,18)*.

```
int f(int a, int b) {
      while (b != 0) {
            int r = a % b;
            a = b;
            b = r;
      }
      return a;
}
```

a) cel mai mare divizor comun, valoarea returnată va fi 6
b) cel mai mare divizor comun, valoarea returnată va fi 18
c) cel mai mic multiplu comun, valoarea returnată va fi 48
d cel mai mic multiplu comun, valoarea returnată va fi 144

14. Ce afișează programul de mai jos?

```
void main(){
	int i, suma = 0;
	for (i = 1; i <= 10; i++)
		if (i % 2 == 0)
			suma += i;
	printf("%d\n", suma);
}
```

a) 20 **b)** 30 **c)** 40 **d)** 50

15. Se consideră subprogramul de mai jos. Ce se va afișa la apelul acestuia? Se consideră că vectorul $v = \{11, 2, 31, 16, 10341\}$ și $n = 5$.

```
void print(int v[], int n){
	int i, d;
	bool flag;
	for (i = 0; i < n; i++)
	{
		flag = true;
		for (d = 2; d * d <= v[i]; d++)
			if (v[i] % d == 0)
				flag = false;

		if (flag)
			printf("%d ", v[i]);
	}
}
```

a) 2 31 10341
b) 11 2 31 16 10341
c) 11 2 31 10341
d) 11 2 31

16. Ce afișează programul de mai jos?

```
void count(int value) {
	int res = 0;
	while (value > 0) {
		if ((value % 10) % 2 != 0)
			res += 1;
		value = value / 10;
	}
	printf("%d", res);
}
int main() {
	count(1719556);
}
```

a) 1 **b)** 7 **c)** 0 **d)** 6

17. Indicați ce va afișa programul de mai jos.

```
void main() {
      int i, s = 0;
      for (i = 0; i < 100; i++)
            if (i % 2 == 0 && i % 3 == 1)
                  s += i;
      printf("%d", s);
}
```

a) Suma numerelor din intervalul [0,99]

b) Suma numerelor pare din intervalul [0,99]

c) Suma numerelor pare din intervalul [0,99] al căror rest în urma împărțirii la 3 este 1.

d) Suma numerelor pare din intervalul [0,99] care sunt divizibile cu 3

18. Se consideră variabila întreagă *a = 12345678*. Care dintre expresiile următoare înlocuiește cifra *5* cu cifra *9*?

a) *a = a / 100 * 100 + 9 * 10 + a % 10*;

b) *a = a / 100000 * 100000 + 9 * 10000 + a % 10000*;

c) *a = a / 1000 * 1000 + 9 * 100 + a % 100*;

d) *a = a / 10000 * 10000 + 9 * 1000 + a % 1000*;

19. Se consideră subprogramul *calcul*, definit mai jos. Ce valoare va fi întoarsă de apelul *calcul(28)?*

```
int calcul(int x) {
      int p = 1, i, j = 1;
      for (i = 2; x != 1; i++)
      {
            if (x % i == 0)
            {
                  while (x % i == 0)
                        x /= i;
                  j *= i;
            }
      }
      return j;
}
```

a) 7 **b)** 2 **c)** 10 **d)** 14

20. Care va fi valoarea stocată în variabila *sum* în urma execuției următoarei secvențe de cod?

```
int n = 199756;
int a = n / 10 % 100 / 2 % 11;
int b = n % (100 / 10);
int sum = a + b;
```

a) 9 **b)** 19 **c)** 10 **d)** 75

21. Ce afișează programul următor?

```
int x, y = 2, z = 3;
void f(int x, int z){
      x += 2;
      y = ++x;
      z = x-- + 2;
      printf("%d %d %d\n", x, y, z);
}
int main(){
      int x = 2;
      f(x, y);
      printf("%d %d %d\n", x, y, z);
      f(y, y);
      printf("%d %d %d\n", x, y, z);
}
```

a) $\begin{pmatrix} 4 & 5 & 7 \\ 2 & 5 & 3 \\ 7 & 8 & 10 \\ 2 & 8 & 3 \end{pmatrix}$ **b)** $\begin{pmatrix} 2 & 3 & 5 \\ 0 & 3 & 3 \\ 5 & 6 & 8 \\ 0 & 6 & 3 \end{pmatrix}$ **c)** $\begin{pmatrix} 2 & 3 & 5 \\ 2 & 3 & 3 \\ 5 & 6 & 8 \\ 2 & 6 & 3 \end{pmatrix}$ **d)** $\begin{pmatrix} 4 & 5 & 7 \\ 0 & 5 & 3 \\ 7 & 8 & 10 \\ 0 & 8 & 3 \end{pmatrix}$

22. Precizați ce variantă conține doar nume de variabile permise pentru utilizarea în limbajul C:

a) var, _CoNtOr, my_value;
b) Value, total-sum, suma;
c) contor, 1_rezultat, valoare;
d) double, maxValue, contor;

23. Fie *a* și *b* două numere naturale. Precizați care dintre următoarele afirmații este adevărată cu privire la valoarea următoarei expresii:

```
((a && !b) && (a && (!b || !a)))
```

a) Expresia are valoarea *1* dacă și numai dacă *a=0* și *b=1*
b) Expresia are valoarea *1* dacă și numai dacă *a=1* și *b=0*
c) Expresia are valoare *1* dacă și numai dacă *a=0* și *b=0*
d) Expresia are valoare *1* dacă și numai dacă *a=1* și *b=1*

24. Ce valoare are variabila *a* după execuția secvenței de cod următoare?

```
int a = 10, b = 11, c = 12, d = 13;
a += b = c = ++d;
/* a++; */
a += 2;
```

a) 23 **b)** 24 **c)** **25** **d)** 26

25. Precizați valoarea variabilei *c* după execuția secvenței de cod următoare:

```
int a = 2527, b = 971208, c = 0;
int val = a / 10 % 10;
while (b)
{
	if (b % 10 > val)
		c++;
	b /= 10;
}
```

a) 1 **b)** 3 **c)** 2 **d)** 4

26. Se consideră subprogramul de mai jos, unde *v* reprezintă un vector de elemente întregi, iar *n* reprezintă numărul de elemente al vectorului *v*. Menționați ce va afișa apelul funcției *print* dacă vectorul *v* conține elementele *{1, 5, 6, -14, 5, 1, 7, -13, 0, 2}*.

```
void print(int v[], int n) {
	int i, r = 0;
	for (i = 0; i < n; i++)
	{
		r += v[i];
		if (v[i] > r)
			printf("%d ", v[i]);
	}
}
```

a) -14 -13 **b)** 5 0 2 **c)** -14 5 -13 0 2 **d)** 6 5 7 0 2

27. Se consideră subprogramul de mai jos. Care este valoarea sumei $\sum_{i=0}^{10} verificare(i)$?

```
int verificare(int n){
	int suma = 0, i;
	for (i = 1; i <= n / 2; i++)
		if (n % i == 0)
			suma += i;
	return suma == n;
}
```

a) 0 **b)** 5 **c)** 1 **d)** 2

28. Precizați care dintre afirmațiile de mai jos este adevărată cu privire la valoarea întoarsă de apelul $prog(m,n)$, dacă m și n sunt numere naturale, n este impar și $m > n + 1$.

```
int prog(int m, int n)
{
	int s = 0, i;

	for (i = 1; i <= m; i += 2)
		if (i % n == 0)
			s += i;

	return s;
}
```

a) O valoare care este întotdeauna mai mare sau egală cu *m*
b) O valoare care este întotdeauna strict mai mică decât *m*
c) O valoare care este întotdeauna plasată în intervalul [*m+n-1, m+2*n*]
d) Niciuna din variantele de mai sus nu este corectă

29. Menționați ce valori pot fi introduse, pe rând, pentru ca programul de mai jos să afișeze 20 de fiecare dată.

```
void main()
{
		int x, y;
		scanf("%d", &x);

		switch (x%3)
		{
		case 0:
			y = x % 10;
			break;

		case 1:
			y = x / 10;
			break;

		default:
			y = 10;
			break;
		}

		printf("%d", x - y);
}
```

a) 22, 24 **b)** 23, 13 **c)** 122, 125 **d)** 32, 33

30. Se consideră subprogramul de mai jos, unde *n* reprezintă un număr întreg din intervalul [10, 10^9]. Menționați ce va returna apelul funcției *print(753465634)*.

```
long print(long n) {
      long p = 1, var = 0;
      while (n / p) {
            int c = n / p % 2;
            var = c * p + var;
            p *= 10;
      }
      return var;
}
```

a) 110101 **b)** 433656437 **c)** 1010010110 **d)** 111001010

31. Menționați ce se va afișa la execuția următoarei secvențe de cod.

```
int i = 0;
for (i = 0; i < 6 && i % 2 == 0; i++)
{
      printf("%d ", i++);
      printf("%d ", --i);
}
```

a) 0 1 2 3 4 5 **b)** 0 0 **c)** 0 1 **d)** 0 0 2 2 4 4

32. Se consideră subprogramul de mai jos, unde *x* și *a* sunt două numere întregi. Menționați ce afișează apelul funcției *func(127,3)*.

```
void func(int x, int a)
{
      while (x > a)
      {
            if (x % a)
            {
                  x = x / a;
                  a += 1;
            }
            else
            {
                  x -= a;
                  a = a * 10;
            }
      }
      printf("%d %d", x, a);
}
```

a) 2 6 **b)** 10 50 **c)** 5 50 **d)** 94 300

33. Indicați valoarea variabilei *a* după execuția secvenței de mai jos:

```
int a = 15;
a = a % (a + 1) / (a % 4 + 1) + a/10/10;
```

a) 3 **b)** 2 **c)** 4 **d)** 1

34. Indicați valorile variabilelor *x*, *y* și *z* obținute în urma execuției următoarei secvențe de instrucțiuni:

```
int x, y, z;
x = 1, y = 0, z = 2;
x += z % 3;
y = x / 2;
z += (x + y) % 2 + z;
```

a) x = 3, y = 1, z = 4
b) x = 2, y = 1, z = 4
c) x = 3, y = 0, z = 2
d) x = 2, y = 0, z = 2

35. Fie funcția *f* având parametrii *n*, *a*, *b* numere naturale, și $n > 5$. Indicați ce verifică funcția *f*, ținând cont de limitările impuse anterior.

```
int f(int n, int a, int b){
      if (n <= 1)
            return 0;
      int i;
      for (i = 2; i * i <= n; i++)
            if ((n % i == 0) || (n < a || n >= b))
                  return 0;
      return 1;
```

a) dacă un număr este palindrom
b) dacă un număr este palindrom și aparține intervalului (a,b]
c) dacă un număr este prim și aparține intervalului (a,b]
d) dacă un număr este prim și aparține intervalului [a,b)

36. Ce putem afirma despre subprogramul de mai jos?

```
void function(int a, int b)
{
      int n, k, j;
      for (n = a; n < b; n++)
      {
            k = 0;
            if (n < 2)
                  k = 1;
            for (j = 2; j < n / 2; j++)
                  if (n % j == 0)
                        k = 1;
            if (k == 0 && (n / 10 % 10 == n % 10))
                  printf("%d\n", n);
      }
}
```

a) afișează toate numerele care nu sunt prime, aparțin intervalului [a,b) și au ultimele 2 cifre identice
b) afișează toate numerele prime care aparțin intervalului [a,b) și au ultimele 2 cifre identice
c) afișează toate numerele prime care aparțin intervalului [a,b) și au primele 2 cifre identice
d) afișează toate numerele prime care aparțin intervalului [a,b] și au primele 2 cifre identice

37. Pentru ce valoare atribuită variabilei *a* programul următor va afișa valoarea *2025*?

```
int main(){
      int a = 8103;
      int b = a % 100;
      while (b > 0 && a > 4000)
      {
            a = a / 2;
            b = b - 2;
      }
      printf("%d", a);
}
```

a) 8103 **b)** 8105 **c)** 16208 **d)** 4052

38. Menționați cu ce condiție trebuie înlocuită linia punctată pentru ca funcția *sunt_prime* să verifice corect dacă *a* și *b* sunt prime între ele.

```
void sunt_prime(int a, int b){
      int min = (a < b) ? a : b;
      int conditie = 0;
      for (int i = 2; i <= min; i++)
            if (.................)
                  conditie = 1;

      if (conditie == 0)
            printf("a si b sunt prime intre ele");
      else
            printf("a si b NU sunt prime intre ele");
}
```

a) `a % i == 0 || b % i == 0`
b) `a / i == 0 && b / i == 0`
c) `a / i == 0 || b / i == 0`
d) `a % i == 0 && b % i == 0`

39. Ce se va afișa în urma executării secvenței de instrucțiuni de mai jos?

```
int v[5] = { 1 }, i;
for (i = 0; i < 5; i++)
{
      switch (v[i])
      {
      case 0: v[i] = 1; break;
      case 1: v[i] = 2; break;
      case 2: v[i] = 3; break;
      default: v[i] = 0; break;
      }
}
for (i = 0; i < 5; i++)
      printf("%d ", v[i]);
```

a) 2 2 2 2 2 **b)** 2 1 1 1 1 **c)** 3 2 2 2 2 **d)** 0 0 0 0 0

40. Fie secvența de instrucțiuni de mai jos. De câte ori se va executa instrucțiunea `x+=2`?

```
int x = 52;
int y = 13;
while (x % y != x / y) {
      x += 2;
      y = y - 1;
}
```

a) 0 **b)** 4 **c)** 9 **d)** 1

41. Indicați expresia care are valoarea *1* dacă și numai dacă $1 \leq x < 3$?

a) `(x >= 1) && (x < 3)`
b) `x * x - 4 * x + 3 <= 0`
c) `(x - 1) * (x - 3) > 0`
d) `(x > 1) || (x < 3)`

42. Indicați ce afișează execuția programului de mai jos.

```
int f(int x){
      x = x * 2;
      return x;
}
void main(){
      int x = 1, y = 1, z = 1;
      y = f(++x);
      z = x * 2;
      printf("%d %d %d", x, y, z);
}
```

a) 2 4 2 **b)** 1 2 2 **c)** 4 4 8 **d)** 2 4 4

43. Precizați ce valoare întoarce apelul funcției *f(1091740)*, pentru funcția *f* definită mai jos.

```
int f(int  a){
      int i, b, x = 0;
      for (i = 2; i < a / 2; i++)
      {
            b = 0;
            while (a % i == 0) {
                  a = a / i;
                  b++;
            }
            if (b != 0)
                  x++;
      }
      if (a != 1)
            x++;
      return x;
}
```

a) 0 **b)** 5 **c)** 7 **d)** 545870

VECTORI ȘI MATRICE

44. Se consideră subprogramul de mai jos. Menționați care este condiția ce trebuie inserată în locul spațiului marcat prin . . . . . astfel încât subprogramul să returneze suma elementelor impare aflate pe poziții pare.

```
int f(int a[], int n){
      int i, s = 0;
      for (i = 0; i < n; i++)
            if (. . . . . .)
                  s += a[i];
      return s;
}
```

a) `i % 2 == 0 && a[i] % 2 == 0`
b) `a[i] % 2 == 1`
c) `i % 2 == 0 % a[i] % 2 != 1`
d) `i % 2 == 0 && a[i] % 2 == 1`

45. Precizați ce valoare conține elementul *T[1][1]* pentru tabloul bidimensional definit mai jos:

```
int T[ ][2] = { 1,2,3,4,5,6 };
```

a) 2 **b)** 3 **c)** 4 **d)** 5

46. Precizați care este șirul de valori afișat în urma execuției programului de mai jos.

```
void main() {
      int N = 4,i,sum=0;
      int A[][4] = {  {10, 8, 6, 4},
                      {11,13,15,17},
                      { 2, 4, 6, 8},
                      {12,14,16,18}};

      for (i = 0; i < N; i++) {
            printf("%d %d ", A[i][i], A[N - i - 1][i]);
      }
}
```

a) 10 13 6 18 4 15 4 12
b) 10 13 6 18 12 4 15 4
c) 4 15 4 12 10 13 6 18
d) 10 12 13 4 6 15 18 4

47. Un magazin online gestionează inventarul folosind doi vectori: *nr_buc* și *preturi*, fiecare de *n* elemente. Vectorul *nr_buc* reține numărul de bucăți disponibile din fiecare produs. Vectorul *preturi* reține prețul unei bucăți din fiecare produs. Vectorii sunt indexați de la *0* la *n−1*. Considerăm variabila *total=0*. Care dintre următoarele secvențe de cod calculează valoarea inventarului (suma totală a prețurilor tuturor produselor din magazin)?

a)
```
for(int i = 0; i < N; i++)
      total += nr_buc[0] * preturi[0];
```

b)
```
for(int i = 0; i < N; i++)
      total += preturi[i];
```

c)
```
for(int i = 0; i < N; i++)
      total += nr_buc[i] + preturi[i-1];
```

d)
```
for(int i = 0; i < N; i++)
      total += nr_buc[i] * preturi[i];
```

48. Se consideră secvența de instrucțiuni de mai jos, unde *a* reprezintă o matrice de elemente întregi, iar *N* numărul de linii și de coloane al matricei. Selectați instrucțiunea corectă care înlocuind linia punctată va face ca matricea a să fie egală cu $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 2 & 4 & 1 & 3 \\ 0 & 3 & 1 & 4 & 2 \\ 0 & 4 & 3 & 2 & 1 \end{pmatrix}$.

```
#define N 5
int a[N][N] = {0}, i, j;
for(i=0 ;i<N;i++)
      for(j=0;j<N;j++)
            ..................
```

a) `a[i][j] = (i*j)%N;`
b) `a[i][j] = (i+j)%N;`
c) `a[i][j] = (i+j)%(N+1);`
d) `a[i][j] = (i*j)%(N+1);`

49. Fie următoarea secvență de instrucțiuni, iar variabila *m* reprezintă o matrice de *10* linii și *10* coloane. Precizați câte valori pare va conține diagonala secundară a matricei *m* în urma execuției secvenței de instrucțiuni.

```
for (int i = 0; i < 10; i++)
      for (int j = 0; j < 10; j++)
      {
            m[i][j] = (2 * j - i) % 10;
            m[i][j] = abs(m[i][j]);
      }
```

a) 4 **b)** 6 **c)** 5 **d)** 10

50. Indicați valorile din matricea *a*, după execuția următoarei secvențe:

```
int i, j, n = 4;
int a[4][4];
for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
            if (i - j)
                  a[i][j] = j + 1;
            else
                  a[i][j] = i + j;
```

a) $\begin{pmatrix} 0 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 4 & 4 \\ 1 & 2 & 3 & 6 \end{pmatrix}$

b) $\begin{pmatrix} 0 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 4 \end{pmatrix}$

c) $\begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 4 & 5 \\ 1 & 2 & 3 & 6 \end{pmatrix}$

d) $\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix}$

51. Fie funcția *f* de mai jos. Precizați cu ce instrucțiune/set de instrucțiuni trebuie înlocuită linia punctată astfel încât funcția să calculeze diferența minimă absolută dintre oricare doi termeni consecutivi ai vectorului *v*.

```
int f(int v[], int n) {
      int diff = 0;
      int min = v[1] - v[0];
      if (min < 0)
            min = -min;
      for (int i = 1; i < n - 1; i++) {
            . . . . . .
            if (min > diff)
                  min = diff;
      }
      return min;
}
```

a)
```
diff = v[i] - v[i + 1];
if (diff < 0)
      diff = -diff;
```

b)
```
diff = v[i+1] - v[i];
```

c)
```
diff = v[i] - v[i + 1];
```

d)
```
diff = v[i] - v[i + 1];
diff = -diff;
```

52. Se consideră programul de mai jos. Ce se afișează pe ecran?

```
void main() {
      int mat[4][4] = {{ 0, 1, 2, 3 },
                       { 4, 5, 6, 7 },
                       { 8, 9,10,11 },
                       {12,13,14,15 }};
      int i = 1, j = 1, sum = 0;
      for (i = 0; i < 4; i++)
            for (j = 0; j < 4; j++)
                  if (i == 0 || j == 0 || i == 3 || j == 3)
                        sum += mat[i][j];
      printf("%d", sum);
}
```

a) 90 **b)** 80 **c)** 119 **d)** 70

53. Fie dată o matrice pătratică *matrix*, având *n* linii și *n* coloane. Completați linia punctată astfel încât secvența de instrucțiuni obținută să înlocuiască toate elementele de pe diagonala secundară și de deasupra acesteia cu valoarea *2*.

```
int i, j;
for (i = 0; i < n; i++)
      . . . . . .
            matrix[i][j] = 2;
```

a) `for (j = 0; j <= n - i - 1; j++)`
b) `for (j = i; j <= n - 1; j++)`
c) `for (j = n - i - 1; j < n; j++)`
d) `for (j = 0; j < i; j++)`

54. Ce calculează programul de mai jos și ce valoare este afișată?

```
void main() {
	int mat[3][3] = {{1, 2, 3},
			 {4, 5, 6},
			 {7, 8, 9}};
	int s = 0, i;
	for (i = 0; i < 3; i++)
		s += mat[i][2 - i];
	printf("%d", s);
}
```

a) suma elementelor de pe diagonala principală și mesajul afișat este 15
b) suma elementelor de pe diagonala secundară și mesajul afișat este 15
c) suma elementelor de sub diagonala principală și mesajul afișat este 19
d) suma elementelor de sub diagonala secundară și mesajul afișat este 23

55. Ce afișează programul de mai jos?

```
void calculeaza(int v[], int n, int x){
	int c = 0;
	for (int i = 0; i < n; i++)
		if (v[i] == x)
			c++;
	printf("%d", c);
}
void main(){
	int v[] = { 1, 2, 3, 4, 3, 2, 1, 3, 3 };
	calculeaza(v, 9, 3);
}
```

a) 2 **b)** 3 **c)** 4 **d)** 5

56. Se consideră programul de mai jos. Considerând că se vor introduce valorile *1*, *2*, *3*, *4*, *5*, ce va afișa programul?

```
void main(){
	int v[10] = { 0 };
	int n=5, aux, i, j = 0;
	for (i = 0; i < n; i++)
	{
		scanf("%d", &aux);
		if (aux % 2 == 0)
			continue;
		v[j++] = aux;
	}
	for (i = 0; i < j; i++)
		printf("%d ", v[i]);
}
```

a) 1 2 3 4 5 **b)** 2 4 **c)** 1 2 3 **d)** 1 3 5

57. Se consideră programul de mai jos. Menționați ce va fi afișat in urma execuției acestuia.

```
void print(int matrix[][5], int n, int m) {
      int i, j;
      for (i = 0; i < n; i++)
            for (j = 0; j < m; j++)
                  if (i % 2 == 0 && j % 2 == 1)
                        printf("%d", matrix[i][j]);
}
void main() {
      int matrix[][5] = {   {1,2,3,4,5},
                            {5,4,3,2,1},
                            {6,7,8,9,0},
                            {0,9,8,7,6},
                            {0,2,4,6,8} };
      print(matrix, 4, 4);
}
```

a) 272496 **b)** 4297 **c)** 247926 **d)** 2479

58. În urma executării secvenței de instrucțiuni de mai jos, produsul elementelor de pe diagonala secundară a matricei *matrix*, cu *4* linii și *4* coloane va fi :

```
int matrix[10][10];
int n = 4;
int i, j;
for (i = 0;i < n; i++)
      for (j = 0;j < n; j++)
            matrix[i][j] = n - i + j;
```

a) 0 **b)** 35 **c)** 256 **d)** 105

59. Menționați ce afișează secvența de program de mai jos.

```
int v[] = { 11, 234, 567, 0, 456, 24 }, s = 0;
for (int i = 0; i < sizeof(v) / sizeof(int); i++)
{
      while (v[i])
      {
            s++;
            v[i] /= 10;
      }
}
printf("%d", s);
```

a) 12 **b)** 13 **c)** 14 **d)** 50

60. Menționați valoarea elementelor matricei *m* la finalul execuției secvenței de instrucțiuni de mai jos.

```
int i, j, m[3][3]={{4,1,8},{2,9,5 },{7,3,6} };
for (i = 0; i < 3; i++)
      for (j = 0; j < 3; j++)
      {
            if (i % 2)
                  ++m[i][j];
            if (!(j % 2))
                  m[i][j] /= 2;
}
```

a) $\begin{pmatrix} 2 & 1 & 4 \\ 1 & 10 & 3 \\ 3 & 3 & 3 \end{pmatrix}$ **b)** $\begin{pmatrix} 2 & 2 & 4 \\ 1 & 9 & 2 \\ 4 & 4 & 3 \end{pmatrix}$ **c)** $\begin{pmatrix} 4 & 0 & 8 \\ 3 & 5 & 6 \\ 7 & 1 & 6 \end{pmatrix}$ **d)** $\begin{pmatrix} 5 & 1 & 9 \\ 2 & 4 & 5 \\ 8 & 2 & 7 \end{pmatrix}$

61. Indicați valoarea variabilei *sum*, după execuția următoarei secvențe:

```
int matrix[50][50] = { 0 };
int i = 0, j = 0, n=49;
for (i = 0; i < n; i++)
{
      matrix[i][i] = -1;
      matrix[i][n - i - 1] = 1;
}
int sum = 0;
for (i = 0; i < n; i++)
{
      for (j = 0; j < n; j++)
            sum += matrix[i][j];
}
```

a) -1 **b)** 1 **c)** 50 **d)** valoare necunoscută

62. Ce va conține vectorul *v* în urma execuției următoarei secvențe de cod?

```
int v[] = { 0,0,0 }, n = 5, i;
for (i = 0; i < n; i++)
{
      if (v[i])
            v[i] += 1;
}
```

a) {1,1,1,1,1}
b) {0,0,0,0,0}
c) {5,0,0,0,0}
d) {0,0,0,0,5}

63. Se consideră programul de mai jos. Precizați care este valoarea elementelor vectorului *v* după execuția buclei *for*.

```
void main()
{
	int status = 1;
	int v[9] = { 0,7,3,2,1,2,5,0,1 };
	int i;
	for (i = 1; i < 9; i += 2)
	{
		if (v[i] % 2 == 0)
			status = (v[i+1] && v[i-1]) || (!status);
		else
			status = v[i-1] || v[i+1];

		if (status == 1)
			v[i++]++;
	}
}
```

a) 0 8 3 3 1 3 5 1 1
b) 0 7 3 2 1 2 5 0 1
c) 0 9 3 4 1 4 5 2 1
d) 0 8 3 2 2 2 5 1 1

64. Se consideră matricea $A = \begin{pmatrix} 2 & 2 & 2 \\ 1 & 0 & 1 \\ 2 & 2 & 2 \end{pmatrix}$ și secvența instrucțiuni de mai jos, unde toate variabilele sunt întregi și *A* este o matrice de *3x3*.

```
for (i = 0; i < 3; i++) {
	for (j = 0; j < 3; j++) {
		if (i % 2 == 0)
			A[i][j] = (a + b) % 3;
		else
			if (j % 2 == 1)
				A[i][j] = b % 3;
			else
				A[i][j] = (a + b - i) % 3;
	}
}
```

Pentru ce valori *a* și *b* se obține matricea A, după executarea secvenței de cod?

a) a = 32; b = 96;
b) a = 31; b = 97;
c) a = 31; b = 96;
d) a = 32; b = 97;

65. Pentru subprogramul de mai jos, alegeți care dintre variante poate reprezenta valoarea șirului s, astfel încât subprogramul să afișeze *1 abcba*.

```
void func(char s[]) {
      int n = strlen(s);
      int count = 0;
      for (int i = 0; i < n / 2; i++) {
            if (s[i] != s[n - i - 1]) {
                  s[n - i - 1] = s[i];
                  count++;
            }
      }
      printf("%d ", count);
      printf("%s\n", s);
}
```

a) abcab **b)** abcxa **c)** abcba **d)** abcdx

66. Indicați ce realizează funcția *print*, definită mai jos. Se consideră *v*, un vector de elemente întregi și *n*, numărul elementelor lui *v*.

```
void print(int v[], int n) {
      int i;
      for (i = 0; i < n; i++)
            if (i % 2)
                  if (v[i] % 2 == 1)
                        printf("%d ", v[i]);
}
```

a) toate elementele pare din vector
b) toate elementele impare din vector
c) elemente impare de pe poziții impare
d) elementele pare de pe poziții impare

67. Fie secvența de cod de mai jos. Indicați ce valori trebuie să fie în vectorul *v* astfel încât după execuția secvenței, valoarea variabilei *x* să fie 50.

```
int x = 10;
for (int i = 0; i < 4; i=i+2)
      x = x + (v[i] / 2) + v[i + 1];
```

a) *13, 17, 9, 13, 23*
b) *21, 14, 1, 22, 10*
c) *21, 14, 8, 22, 10*
d) *13, 17, 19, 13, 83*

68. Fie o matrice *B* cu *N* linii și *N* coloane, unde *N* este un număr întreg pozitiv mai mare decât *1*. Fiecare element *B[i][j]* (unde $0 \leq i < N$ și $0 \leq j < N$) este definit prin relația:

$$B[i][j] = (i+1)^3 + (j+1)^3$$

Care este valoarea ultimului element de pe diagonala secundară a matricei?

a) $(N)^3 + 1$ **b)** $(N)^3 + 2^3$ **c)** $(N)^3 + (N-1)^3$ **d)** $2N^3$

69. Considerând funcția de mai jos, ce valori va conține matricea *a* în urma apelului *generateMatrix(4, a)*?

```
void generateMatrix(int n, int a[4][4]) {
     int i, j;
     for (i = 0; i < n; i++)
          for (j = 0; j < n; j++){
               a[i][j] = i * n + j * n;
               if (a[i][j] * j % 2 == 0)
                    a[j][i] = 4;
          }
}
```

a) $\begin{pmatrix} 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 \\ 8 & 12 & 4 & 4 \\ 12 & 16 & 20 & 4 \end{pmatrix}$

b) $\begin{pmatrix} 4 & 4 & 4 & 4 \\ 4 & 8 & 12 & 16 \\ 8 & 12 & 4 & 4 \\ 12 & 16 & 20 & 24 \end{pmatrix}$

c) $\begin{pmatrix} 4 & 4 & 4 & 4 \\ 16 & 20 & 24 & 28 \\ 32 & 40 & 4 & 4 \\ 48 & 60 & 72 & 84 \end{pmatrix}$

d) Niciuna dintre variantele de mai sus.

70. Precizați care afirmație este corect verificată de funcția de mai jos (afișează mesajul *result: true*) considerând că *V* este un vector oarecare de dimensiune n:

```
void check(int V[ ], int n){
      int i;
      for (i = n/2 - 1; i >= 0; i--)
            if (V[i] != V[n - i - 1])
                  break;
      if (i + 1 == 0)
            printf("result: true");
      else
            printf("result: false");
}
```

a) toate elementele vectorului au aceeași valoare
b) cel puțin jumătate din elementele vectorului au valoarea diferită de 0
c) cel puțin jumătate din elementele vectorului sunt diferite între ele
d) vectorul este palindrom

71. Se consideră subprogramul de mai jos, care face parte dintr-un program de criptare a șirului de caractere *mesaj*. Funcția *fun* implementează primul pas al programului - completarea unei matrice pătratice conform unei reguli. Care este lungimea maximă a șirului de caractere care va putea fi criptat, astfel încât să nu fie pierdute informații din mesajul inițial?

```
void fun(char m[50][50], char mesaj[]) {
      int i, j, k=0;
      for(i=0;i<50;i++)
            for (j = 0; j < 50; j++) {
                  if (j >= i)
                        m[i][j] = mesaj[k++];
                  if (mesaj[k] == 0)
                        break;
            }
}
```

a) 1400 **b)** 2500 **c)** 1275 **d)** 100

72. Se consideră funcția *isPrime* având prototipul *int isPrime(int n)*, care întoarce *1* dacă *n* este prim și *0* altfel. Indicați care dintre următoarele matrice *m* ar face ca apelul *verif(m,4)* să returneze *1*.

```
int verif(int m[4][4], int N){
      int i, sum = 0;
      for (i = 0; i < N; i++)
            sum += m[i][N - i - 1];
      return isPrime(sum);
}
```

a) $\begin{pmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{pmatrix}$ **b)** $\begin{pmatrix}0&1&1&2\\1&0&3&1\\1&0&0&0\\1&1&0&0\end{pmatrix}$ **c)** $\begin{pmatrix}1&1&1&1\\1&2&2&1\\1&2&2&0\\1&1&0&0\end{pmatrix}$ **d)** $\begin{pmatrix}0&1&1&1\\1&0&0&1\\1&0&0&0\\1&1&0&0\end{pmatrix}$

73. Se consideră subprogramul de mai jos, unde *v* reprezintă un vector de elemente întregi, iar *n* numărul de elemente al vectorului *v*, *si* funcția *max* care întoarce elementul cu valoarea absolută cea mai mare din vectorul *v*. Menționați ce va afișa apelul funcției *print*.

```
void print(int v[], int n){
      int i, j, mx = max(v,n);
      for (i = 0; i < n; i++)
            for (j = 0; j * j < mx; j++)
            {
                  if (v[i] < 0)
                        break;
                  if (!(v[i] - j * j))
                        printf("%d", v[i]);
            }
}
```

a) Toate elementele negative din vector
b) Toate elementele pătrate perfecte in valoare absoluta
c) Toate elementele pozitive
d) Toate elementele nenegative pătrate perfecte

74. Se consideră subprogramul de mai jos, unde *v* este un vector de numere întregi, iar *n* este număr întreg, dimensiunea vectorului. Menționați ce va afișa apelul funcției *calculeaza* pentru *v = {4, -1, 2, 1, -5, 4}* și *n* = 6.

```
void calculeaza(int v[], int n) {
      int s = 0;
      for (int i = 0; i < n; i++) {
            if (v[i] > 0) {
                  s += v[i];
                  printf("%d ", v[i]);
            }
      }
      printf("%d", s);
}
```

a) 8 **b)** 1 -5 4 9 **c)** 4 -1 2 1 10 **d)** 4 2 1 4 11

75. Precizați ce afișează programul de mai jos.

```
void main()
{
	int N = 5,i,sum=0;

	int A[][5] = {  {9,8,7,6,5},
			{2,4,6,8,0},
			{1,3,5,7,8},
			{1,2,3,4,5},
			{4,5,6,7,8} };

	for (i = 0; i < N; i += 2)
		sum += (A[i][i] + A[N - i - 1][i]);

	printf("%d", sum);
}
```

a) 36 **b)** 54 **c)** 39 **d)** 28

76. Precizați care este valoarea șirului de caractere *s* la finalul executării următoarei secvențe de program.

```
char f(char x)
{
	return x++;
}
void main()
{
	char s[] = "abcde";
	int i = 0;

	while (s[i + 1])
	{
		s[i] = f(s[i + 1]);
		i++;
	}
}
```

a) bcdef **b)** bcdee **c)** cdefe **d)** abcde

77. Precizați ce se afișează la execuția funcției *func* definită mai jos când este apelată pentru o matrice *mat* pătratică de dimensiune *n*.

```
void func(int mat[100][100], int n){
      int i, j;
      for (i = 0; i < n; i++)
            for (j = i + 1; j < n; j++)
                  if (!(mat[j][i]%2) && mat[j][i]%5 == 0)
                        printf("%d ", mat[j][i]);
}
```

a) Elementele divizibile cu 2 și cu 5, aflate deasupra diagonalei secundare a matricei
b) Elementele impare divizibile cu 5, aflate deasupra diagonalei principale a matricei transpuse
c) Elemente divizibile cu 10, aflate deasupra diagonalei principale a matricei transpuse
d) Elementele impare divizibile cu 5, aflate deasupra diagonalei secundare a matricei

78. Care dintre afirmațiile de mai jos este adevărată în cazul programului următor:

```
void main(){
      int mat[4][4], i, j;

      for (i = 0; i < 4; i++)
            mat[0][i] = mat[i][0] = 2;

      for (i = 1; i < 4; i++)
            for (j = 1; j < 4; j++)
                  mat[i][j] = mat[i-1][j] + mat[i][j - 1];
}
```

a) Transpusa matricei generată *mat* conține pe ultima linie și ultima coloană doar valori egale cu 2
b) Matricea generată *mat* conține pe diagonala principală cel puțin 2 numere prime
c) Matricea generată *mat* este simetrică față de diagonala principală
d) Matricea generată *mat* conține cel puțin 2 numere impare strict aflate sub diagonala secundară

79. Precizați ce valori au elementele matricei *mat* după execuția funcției *my_func*:

```
int mat[3][3] = { {11,12,13}, {41,54,61} , {17,18,19} };

void my_func(){
      int i, j, aux, n = 3;
      for (i = 0; i < n; i++)
            for (j = 0; j < n; j++)
                  if (i + j == n - 1)
                  {
                        aux = mat[i][i];
                        mat[i][i] = mat[i][n - i - 1];
                        mat[i][n - i - 1] = aux;
                  }
}
```

a) $\begin{pmatrix} 11 & 12 & 17 \\ 41 & 54 & 61 \\ 13 & 18 & 19 \end{pmatrix}$

b) $\begin{pmatrix} 13 & 12 & 11 \\ 41 & 54 & 61 \\ 19 & 18 & 17 \end{pmatrix}$

c) $\begin{pmatrix} 13 & 12 & 13 \\ 41 & 54 & 61 \\ 17 & 18 & 17 \end{pmatrix}$

d) $\begin{pmatrix} 17 & 12 & 13 \\ 41 & 54 & 61 \\ 17 & 18 & 13 \end{pmatrix}$

ȘIRURI DE CARACTERE

80. Menționați ce se va afișa la execuția următoarelor instrucțiuni:

```
char sir[] = "Academia Tehnica Militara";
printf("%s", strstr(sir, "Militara"));
```

a) Militara
b) Academia Tehnica Militara
c) Tehnica Militara
d) Niciuna din variantele de mai sus

81. Care va fi valoarea șirului *s1* în urma execuției secvenței de mai jos?

```
char s1[100] = "AdmisATM",aux[100];
strcpy(aux, s1 + 2);
strcat(s1, aux + 3);
strcpy(s1, s1 + 5);
```

a) misATMMMMM
b) isATM
c) ATMATM
d) ATMMMMM

82. Ce afișează programul de mai jos?

```
void main() {
      char s[]="Acesta este un exemplu de program";
      int i, j = 0;
      for (i = 0; s[i] != '\0'; i++)
            if (s[i] != ' ')
                  s[j++] = s[i];
      s[j] = '\0';
      printf(s);
}
```

a) \0
b) Acestaesteunexempludeprogram
c) aCESTA ESTE UN PROGRAM
d) ACESTA ESTE UN EXEMPLU DE PROGRAM

83. Indicați cum va fi modificat șirul *sir* în urma apelului funcției definite mai jos.

```
void procesare(char sir[],char ch) {
      while (strchr(sir, ch))
            strcpy(strchr(sir, ch), strchr(sir, ch) + 1);
}
```

a) Funcția intră într-o buclă infinită
b) Oricare ar fi valorile celor două, șirul rămâne nemodificat
c) Sunt eliminate toate aparițiile caracterului *ch* din șirul *sir*
d) Sunt duplicate toate aparițiile caracterului *ch* în șirul *sir*

84. Se consideră programul de mai jos. Ce se afișează pe ecran?

```
void main() {
      char s[]="matematica, fizica, informatica, sport.";
      int i = 0;
      while (1) {
            if (s[i] == 0)
                  break;
            else
                  i++;
      }
      printf("%d", i);
}
```

a) 39 **b)** 1 **c)** 3 **d)** 34

85. Ce afișează programul de mai jos?

```
void main(){
      char sir[] = "Acesta,este:un.sir.";
      int i;
      for (i = 0; sir[i] != 0; i++)
      {
            switch (sir[i])
            {
            case '.':
                  if (sir[i + 1] != 0)
                        sir[i] = ' ';
                  break;
            case ':':
                  sir[i] = ' ';
                  if (sir[i + 1] == 0)
                        sir[i] = '.';
                  break;
            }
      }
      printf("%s", sir);
}
```

a) Acesta,este un sir.
b) Acesta este un sir.
c) Acesta este.un sir.
d) Acesta.este un sir.

86. Precizați ce valoare întoarce apelul *f("1234567890", 3)*:

```
int f(const char* sir, int n){
	int s = 0, i, k;

	for (i = 0; sir[i] != '\0'; i++)
	{
		k = sir[i] - '0';
		if (k % n == 0)
			s += k;
	}
	return s;
}
```

a) 90 **b)** 45 **c)** 18 **d)** 0

87. Precizați ce afișează programul de mai jos.

```
void main(){
	char cuvant [] = "12345abc67890";
	int i, c;
	for (i = 0; cuvant[i] != '\0'; i++){
		if (cuvant[i] >= '0' && cuvant[i] < '9')
		{
			c = cuvant[i] - '0';
			if (c % 2 != 0)
				cuvant[i] = 'a' + (c / 2);
		}
	}
	printf("%s", test);
}
```

a) a2b4cabc6d8e0
b) 1b3c5abcd7e9a
c) a2b4cabc6d890
d) a2b4cybz6d8e0

88. Ce valoare va afișa secvența de instrucțiuni de mai jos?

```
int i;
char v[] = "abracadabra";
for (i = 0; strchr(v, 'a') != 0; i++)
	strcpy(strchr(v, 'a'), strchr(v, 'a') + 1);
printf("%d", i);
```

a) 2 **b)** 0 **c)** 5 **d)** 7

89. Se consideră programul de mai jos. Ce afișează acesta pe ecran?

```
void main() {
      char text[] = "Acesta Este un text.";
      unsigned int L;
      int i;
      L = strlen(text);
      for (i = 1; i < L; i++)
            if (text[i] >= 'A' && text[i] <= 'Z')
                  text[i] = text[i] - 'A' + 'a';

      printf("%s", text);
}
```

a) Acesta Este un text
b) Acesta este un text
c) ACESTA ESTE UN TEXT
d) acesta este un text.

90. Considerând programul de mai jos, ce se va afișa la al 5-lea apel al funcției *printf ?*

```
void f(char s[], int i) {
      if (I < strlen(s)-1) {
            strcpy(s + i, s+i+1);
            printf("%s\n", s);
            f(s, i+1);
      }
}
void main() {
      char str[] = "admitere_atmFI";
      printf("%s\n", str);
      f(str, 0);
}
```

a) admiere_atmFI **b)** ere_atmFI **c)** amtr_tF **d)** diee_atmFI

91. Care este conținutul șirului de caractere *sir*, după execuția secvenței de instrucțiuni de mai jos?

```
char sir[] = "abcdef";
int i;
for (i = 1; i < strlen(sir) - 1; i += 2)
      if (sir[i] >= 'a' && sir[i] <= 'z')
            sir[i - 1] = sir[i];
```

a) bbddef **b)** bbddee **c)** abcdef **d)** aaaaaa

92. Indicați ce afișează programul de mai jos.

```
void main()
{
	char s[] = "Admitere2024ATM";
	int n = strlen(s);
	int count = 0;
	for (int i = 0; i < n; i++) {
		if (s[i] >= 'a' && s[i] <= 'z') {
			count += 1;
		}
		else if (s[i] >= 'A' && s[i] <= 'Z') {
			count += 2;
		}
		else {
			count += 3;
		}
	}
	printf("%d", count);
}
```

a) 32 **b)** 27 **c)** 30 **d)** 24

93. Cu ce condiție trebuie înlocuită linia punctată, astfel încât acesta să afișeze câte perechi de două vocale alăturate avem în șirul *sir*?

```
int main() {
	char sir[] = "A fost odata ca niciodata!";
	char voc[] = "aeiouAEIOU";
	int count = 0;

	for (int i = 1; i < strlen(sir); i++)
		if(strchr(voc,sir[i])&&strchr(voc,sir[i-1]))
			count++;

	printf("%d", count);
}
```

a) `strchr(vocale, sir[i]) && strchr(vocale, sir[i-1])`
b) `sir[i] == vocale && sir[i-1] == vocale`
c) `sir[i] == vocale[i] && sir[i] == vocale[i-1]`
d) `strchr(sir, vocale[i]) && strchr(sir, vocale[i-1])`

94. Pentru care dintre următoarele valori ale variabilei control, următorul program va afișa *Salut George*?

```
void main() {
      char str[100] = "Hello ";
      char control[] = . . . . . .;
      for (int i = 0; i < strlen(control); i++) {
            if (control[i] == 'R')
                  strcpy(str, "Salut ");
            if (control[i] == 'r')
                  strcat(str, "Salut ");
            if (control[i] == 'y')
                  strcat(str, "Ion ");
            if (control[i] == 'G')
                  strcpy(str, "George ");
            if (control[i] == 'g')
                  strcat(str, "George ");
            printf("%s", str);
      }
}
```

a) *”g”* **b)** *”yyMrgx”* **c)** *”rgRG”* **d)** *„mgryRxg”*

95. Fie funcția *f* de mai jos. Ce va afișa apelul *f(24, 3, "academie")*?

```
void f(int x, int y, const char str[]) {
      char my_str[100];
      strcpy(my_str, str);
      for (int i = 0; i < strlen(str); i++)
            if (x % (y + i) == 0)
                  my_str[i] = '*';
      printf("%s\n", my_str);
}
```

a) academie **b)** *c*d*m*e **c)** **a*e*ie **d)** *ca*em*e

96. Precizați ce se afișează în urma execuției programului următor

```
void main(){
      char test1[12] = "Hello";
      char test2[12] = "AtmFI";
      strcpy(test1 + 2, test2);
      strcpy(test2 + 3, test1);
      printf(" %s  %s", test1, test2);
}
```

a) HeAtmFI AtHelAtmFI
b) HeAtmFI AtmHeAtmFI
c) HeAtHello AtHello
d) HeAtmHello AtmHello

97. Se consideră subprogramul de mai jos. Menționați ce va afișa apelul funcției *print* pentru șirul *s="ana Are mEre si pEre"*.

```
void print(char s[100]){
      int i, counter = 0, n=strlen(s);
      for (i = 0; i < n; i++)
            if (!strchr("AEIOUaeiou", s[i]))
                  counter = counter + 1;
      printf("%d", counter);
}
```

a) 12 **b)** 9 **c)** 11 **d)** 8

98. Se consideră subprogramul de mai jos. Menționați ce va afișa apelul funcției *print* pentru șirul *s="Ana ,ArE mEre. si pere!"*.

```
void print(char s[100]){
      int i, counter=0, n = strlen(s);
      for (i = 0; i < n; i++)
            if (!strchr("!?,. ", s[i]))
                  counter+=2;
      printf("%d", counter/2);
}
```

a) 7 **b)** 8 **c)** 24 **d)** 16

99. Se consideră subprogramul de mai jos, unde *s* reprezintă un șir de caractere. Menționați ce va afișa apelul funcției *print pentru șirul "M.A.P.S."*.

```
void f(char s[100]){
      int i, n = strlen(s);
      for (i = 0; i < n; i++)
      {
            if (s[i] == 'S')
            {
                  printf("N");
                  i++;
            }
            if (strchr(",.!?", s[i]))
                  i++;
            printf("%c", s[i]);
      }
}
```

a) M.A.P.S **b)** M.A.P.N **c)** MAPS **d)** MAPN

100. Precizați ce afișează subprogramul de mai jos, în urma apelului *f(s)*, atunci când variabila *s* memorează inițial șirul de caractere *admitereATMFI*:

```
void f(char s[100]){
      char aux[100] = "";
      strcpy(aux, strchr(s, 'A'));
      s[8] = NULL;
      strcpy(aux + 3, aux + 5);
      strcat(aux, "-");
      strcpy(aux + strlen(aux), s);
      printf("%s", aux);
}
```

a) admitere-ATM
b) ATMFI-admitere
c) admitere-ATMFI
d) ATM-admitere

101. Precizați ce afișează următoarea secvență de program:

```
int v[10] = { 0, 1, 1, 1, 2, 2, 2, 3, 3, 3 };
char s[100] = "ana124are27mere389";
for (int i = 0; i < strlen(s); i++)
      if (s[i] >= '0' && s[i] <= '9')
            printf("%d", v[s[i] - '0']);
```

a) 01112133
b) 1121313
c) 11213133
d) 11223321

102. Indicați ce afișează execuția programului următor:

```
void main() {
      int i = 0, c = 3;
      char s[50] = "PROGRAMAREINLIMBAJULC";
      int l = strlen(s);
      for (i = 0; i < l / 2; i += 1) {
            char a = s[i];
            s[i] = s[l - 1 - i];
            s[l - 1 - i] = a;
      }
      printf("%s", s);
}
```

a) PROGRAMAREINLIMBAJULC
b) CLUJABMILNIERAMARGORP
c) CJARGMRINLIAAMBAORPLU
d) CJARGMRINLIAAMBAOPPLU

ALGORITMICĂ ȘI ANALIZA COMPLEXITĂȚII

103. Complexitatea determinării elementului maxim dintr-un vector sortat de *n* elemente este:

a) $O(n)$
b) $max(O(2^n), O(n))$
c) $O(n \cdot log(n))$
d) $O(1)$

104. Care este complexitatea următorului fragment de cod?

```
for (i = 0;i < n;i *= 2)
      v[i]=i+1;
for (i = 0;i < n;i++)
      v[i]++;
```

a) $O(n)$ **b)** $O(2^n)$ **c)** $O(n\,log(n))$ **d)** $O(1)$

105. Precizați ce complexitate are algoritmul implementat de subprogramul *rec*, definit mai jos.

```
int rec(int v[], int N, int i) {
      if (i >= N)
            return 0;
      return v[i] * rec(v, N, i + 1);
}
```

a) $0(N^2)$ **b)** $0(N^3)$ **c)** $0(N)$ **d)** $0(log(N))$

106. Se consideră o matrice pătratică A de dimensiune $n * n$ și o funcție *bubble_sort(int A[], int i)* care sortează elementele de pe linia *i* ale matricei *A* folosind metoda bulelor. Care este complexitatea temporală a următorului subprogram?

```
int i=0;
for(i=0;i<n;i+=2){
      bubble_sort(A,i)
}
```

a) $O(n^2)$ **b)** $O(\frac{n}{2})$ **c)** $O(n^3)$ **d)** $O(log^2(n))$

107. În care dintre următorii vectori putem căuta o valoare folosind algoritmul de căutare binară?

a) 1, 14, 7, 21, 19, 3
b) 1, 7, 33, 40, 54
c) 11, 23, 21, 14, 10
d) 12, 24, 9, 2, 22

108. Se consideră subprogramul de mai jos. Ce va afișa apelul paranteza(“(()(()))(()())((()))()(()(()))”)?

```
void paranteza(char sir[]) {
      int i, deschise = 0, corecte = 0;
      for (i = 0; sir[i] != 0; i++)
      {
            if (sir[i] == ')' && deschise > 0){
                  corecte++;
                  deschise--;
            }
            if (sir[i] == '('
                  deschise++;
      }
      printf("%d", corecte);
}
```

a) 7 **b)** 12 **c)** 14 **d)** 15

109. Fie subprogramul de mai jos. Indicați efectul acestuia atunci când parametrul *nr* este un vector sortat crescător și *nrCifre* este numărul de elemente al vectorului *nr*.

```
void func(int nr[], int nrCifre){
      int i, j, aux;
      for (j = 0; j < nrCifre - 1; j++)
            for (i = 0; i < nrCifre - 1 - j; i++)
            {
                  aux = nr[i];
                  nr[i] = nr[i + 1];
                  nr[i + 1] = aux;
            }
}
```

a) ordonează crescător vectorul
b) ordonează descrescător vectorul
c) interschimbă primul element cu al doilea element
d) nu modifică elementele vectorului

110. Se consideră algoritmul de sortare crescătoare prin metoda selecției (Selection Sort) aplicat la secvența de numere *6, 9, 3, 7, 8, 2*. Care este secvența obținută după a doua parcurgere?

a) 2 9 3 7 8 6
b) 2 3 6 7 8 9
c) 2 3 9 7 8 6
d) 2 3 7 6 8 9

111. Se consideră programul de mai jos. Precizați ce se va afișa în urma lansării în execuție a acestuia.

```
int func(int arr[], int st, int dr, int val){
      if (st > dr)
            return -1;

      int m = st + (dr - st) / 2;
      printf("%d ", arr[m]);
      if (arr[m] == val)
            return m;
      if (arr[m] > val)
            return func(arr, st, m - 1, val);
      else
            return func(arr, m + 1, dr, val);
}
void main() {
      int arr[] = { 1,2,3,4,5,6,7 };
      int rezultat = func(arr, 0, 7, 5);
}
```

a) 4 6 6 **b)** 4 6 5 **c)** 4 7 6 **d)** 4 5 6

112. Ce operațiune realizează subprogramul *f* asupra unui vector?

```
void f(int v[], int n) {
      int i, j, min_index, temp;
      i = 0;
      while (i < n) {
            j = i + 1;
            min_index = i;
            while (j < n) {
                  if (v[j] < v[min_index])
                        min_index = j;
                  j++;
            }
            if (min_index != i) {
                  temp = v[i];
                  v[i] = v[min_index];
                  v[min_index] = temp;
            }
            i++;
      }
}
```

a) Inversează elementele vectorului.
b) Calculează suma elementelor din vector.
c) Ordonează crescător elementele vectorului.
d) Elimina duplicatele din vector.

113. Să se înlocuiască zona marcată prin . . . . . cu expresia corectă, astfel încât subprogramul *fun* să sorteze crescător vectorul *v*.

```
void fun(int v[], int n) {
      int i;
      for (i = 1; i < n; ++i) {
            int k = v[i];
            int j = i - 1;
            . . . . . {
                  v[j + 1] = v[j];
                  j = j - 1;
            }
            v[j + 1] = k;
      }
}
```

a) `while (j >= 0 && v[j] > k)`
b) `while (j > 0 && v[j] > k)`
c) `while (j >= 0 || v[j] > k)`
d) `while (j >= 0 && v[j] < k)`

114. Se consideră subprogramul de mai jos, unde *M* reprezintă o matrice care conține pe fiecare linie un interval de numere întregi delimitat la stânga de elementul de pe prima coloană și la dreapta de elementul de pe cea de-a doua coloană, iar *n* este numărul de intervale. Precizați ce interval va fi afișat pentru următorul vector de intervale: [{1, 7}, {2, 9}, {0, 3}] și *n=3*.

```
void ex(int M[][2], int n){
      int st = M[0][0];
      int dr = M[0][1];
      int i;
      for (i = 1; i < n; i++)
      {
            if (M[i][0] > st && M[i][0] < dr)
                  st = M[i][0];
            if (M[i][1] < dr && M[i][1] > st)
                  dr = M[i][1];
      }
      printf("[%d, %d]\n", st, dr);
}
```

a) [1, 9] **b)** [2, 7] **c)** [0, 3] **d)** [2, 3]

115. Se consideră subprogramul de mai jos. Care este rezultatul apelului acestuia? Se consideră că numerele din vectorul *v* sunt mereu pozitive.

```
void print(int v[], int n){
      int k = -1;
      int index = 0;
      int i, j;
      for (i = 0; i < n; i++)
      {
            for (j = 0; j < n; j++)
            {
                  if (v[j] > k)
                  {
                        k = v[j];
                        index = j;
                  }
            }
            printf("%d ", k);
            k = -1;
            v[index] = -1;
      }
}
```

a) Numerele în ordine aleatorie
b) Numerele în ordine crescătoare
c) Numerele în ordine descrescătoare
d) Numerele în ordinea inversă în care apar în vectorul *v*

116. Se consideră subprogramul de mai jos. Care este rezultatul apelului acestuia? Se consideră vectorul v = {*21*, *22*, *23*, *320*, *2025*}, *n = 5* și *x = 22*.

```
int f(int v[], int n, int x){
      int a = 0, b = n-1;
      while (a <= b)
      {
            int m = (a + b) / 2;
            if (v[m] < x)
                  a = m + 1;
            if (v[m] > x)
                  b = m - 1;
            if (v[m] == x)
                  return m;
      }
      return -1;
}
```

a) –1 **b)** 0 **c)** 1 **d)** 22

117. Menționați ce va afișa secvența de cod de mai jos.

```
int v[] = { 11, 234, 567, 0, 456, 24 }, n = 6, t;
for (int i = 0; i < n; i++)
{
      t = v[i] + v[n - i - 1];
      v[i] = t - v[i];
      v[n - i - 1] = t - v[i];
}

for (int i = 0; i < n; i++)
      printf("%d ", v[i]);
```

a) 24 456 0 567 234 11
b) 24 234 0 567 456 11
c) 24 234 567 0 456 24
d) 11 234 567 0 456 24

118. Vectorul v conține n numere întregi ordonate crescător, iar fragmentul de cod următor își propune să determine dacă există 2 valori distincte în cadrul vectorului v a căror diferență este un număr strict pozitiv *S* dat. Menționați forma corectă pentru expresia lipsă din instrucțiunea if:

```
int i = 0, j=1;
while (j < n)
{
      if (. . . . . .)
            j++;
      else
            if (v[j] - v[i] == S)
                  break;
            else
                  i++;
}
if (j < n)
      printf("yes\n");
else
      printf("no\n");
```

a) `v[i] - v[j] < S`
b) `v[i] - v[j] > S`
c) `v[j] - v[i] < S`
d) `v[j] - v[i] > S`

119. Se dă un tablou bidimensional de numere naturale cu *n* linii și *m* coloane. Fiecare linie a tabloului conține obligatoriu un singur număr impar, celelalte elemente de pe linie fiind numere pare.

Care este complexitatea minimă pentru a determina dacă suma tuturor elementelor tabloului este număr par sau impar?

a) $O(1)$ **b)** $O(n)$ **c)** $O(n * m)$ **d)** $O(n^2)$

120. Se consideră subprogramul de mai jos, unde *v* si *w* sunt doi vectori de numere întregi sortați crescător. *n* și *m* reprezintă dimensiunea vectorului *v*, respectiv *w*. Ce complexitate temporală va avea subprogramul de mai jos?

```
void f(int v[], int n, int w[], int m){
      int i = 0, j = 0, k = 0;
      int rez[1000];

      while (i < n && j < m)
      {
            if (v[i] < w[j])
                  rez[k++] = v[i++];
            else
                  rez[k++] = w[j++];
      }
      while (i < n)
            rez[k++] = v[i++];
      while (j < m)
            rez[k++] = w[j++];
}
```

a) $O(m + n)$ **b)** $O(m * n)$ **c)** $O(m - n)$ **d)** $O(\min(n, m))$

121. Se consideră următorul subprogram care generează toate perechile distincte de elemente dintr-un vector. Care este complexitatea în timp a acestui algoritm pentru un vector de lungime *n*?

```
void genereazaPerechi(int v[], int n) {
      for (int i = 0; i < n - 1; i++) {
            for (int j = i + 1; j < n; j++) {
                  printf("(%d, %d)\n", v[i], v[j]);
            }
      }
}
```

a) $O(n)$ **b)** $O(log\, n)$ **c)** $O(n^2)$ **d)** $O(\sqrt{n})$

122. Se consideră secvența de program de mai jos, unde *A, B și res* sunt matrice de elemente întregi, de dimensiune *nxn*. Menționați care este complexitatea temporală a secvenței.

```
for (i = 0; i < n; i++) {
      for (j = 0; j < n; j++) {
            for (k = 1; k < n; k *= 2)
                  res[i][j] += A[i][k] * B[k][j];
```

a) $O(n^2\,log(n))$ **b)** $O(n^3)$ **c)** $O(n\,log(n))$ **d)** $O(n^2)$

123. Determinați complexitatea de execuție a următorului algoritm:

```
for (i = 0;i < n; i++){
      for (j = 1;j < m; j *= 2){
            aux = var_1;
            var_1 = var_2;
            var_2 = aux;
      }
}
```

a) $O(n*m)$
b) $O(log(n))$
c) $O(m*log(n))$
d) $O(n*log(m))$

124. Care dintre următorii algoritmi are cea mai redusă complexitate pentru o implementare eficientă?

a) QuickSort
b) Interclasarea vectorilor sortați
c) Determinarea submulțimilor unei mulțimi
d) Adunarea a două matrice pătratice cu n linii și n coloane

125. Complexitatea determinării elementului maxim dintr-un vector nesortat de *n* elemente este:

a) $O(n)$ **b)** $max(O(2^n), O(n))$ **c)** $O(n\cdot log(n))$ **d)** $O(1)$

126. Considerând variabilele întregi n, s declarate, ce complexitate are următoarea secvență de cod?

```
int s = 0, i, j;
for (i = 1; i <= n; i++)    {
      for (j = 1; j <= n; j *= 2)
            s++;
      i = j;
}
```

a) $O(log_2(n))$ **b)** $O(n\,log_2(n))$ **c)** $O(n)$ **d)** $O(n^2)$

FUNCȚII RECURSIVE

127. Se consideră subprogramul de mai jos, unde *n* reprezintă un întreg. Menționați ce calculează funcția *f*.

```
int f(int n){
      if (n <= 0)
            return 0;
      return f(n - 1) + n % 2;
}
```

a) Suma ultimelor cifre ale numerelor care sunt in intervalul [*0,n*]
b) Numărul de numere pare care sunt în intervalul [*0,n*]
c) Numărul de numere impare care sunt în intervalul [*0,n*]
d) Suma ultimelor cifre pare ale numerelor care sunt în intervalul [*0,n*]

128. Precizați ce afișează programul de mai jos.

```
int f(int A[][5], int n, int i) {
      if (i == n)
            return 0;
      return
            f(A, n, i + 1) + A[i][n - i - 1] + A[i][i];
}
void main() {
      int A[5][5] = { { 2, 9, 1, 8,12 },
                      { 3, 6, 7, 0,21 },
                      { 1, 4, 6, 2, 5},
                      { 9, 5, 2, 8, 5},
                      { 1, 0, 5, 4, 7} };

      printf("%d", f(A, 5, 0));
}
```

a) 53 **b)** 49 **c)** 66 **d)** 20

129. Se consideră următorul subprogram, unde *b* și *e* sunt două numere întregi. Ce va returna apelul *p(2,11)*.

```
int p(int b, int e) {
      if (e == 0)
            return 1;
      return b * p(b, e - 1);
}
```

a) 2 **b)** 28 **c)** 2048 **d)** 4048

130. Un student are un buget de 10000 lei. El dorește să cheltuiască în prima lună *23* lei, iar apoi să-și dubleze lunar valoarea cheltuielilor (în a doua lună să cheltuiască *46* de lei, în a treia *92* lei și așa mai departe). Studentul implementează o funcție pentru a afla peste câte luni nu va mai avea suficienți bani pentru a continua această metodă de a cheltui bani. Completați spațiile punctate din codul de mai jos astfel încât funcția să-și îndeplinească în mod corect scopul. Funcția este inițial apelată astfel: *f(1, 10000, 23)*.

```
void f(int luna, int buget_ramas, int cheltuiala)
{
      if (buget_ramas < cheltuiala)
            printf("Fonduri terminate in luna %d",luna-1);
      else
            . . . . .
}
```

a) `f(luna + 1, buget_ramas/2 - cheltuiala, cheltuiala);`
b) `f(luna + 1, buget_ramas - cheltuiala * 2, cheltuiala);`
c) `f(luna + 1, buget_ramas - cheltuiala, 2 * cheltuiala);`
d) `f(luna, buget_ramas - cheltuiala, 2 * cheltuiala);`

131. Fie funcția *f* și un șir de caractere *S1=”atm2025”*. Indicați ce se va afișa pe ecran la apelul *f(S1,0)*.

```
void f(char s[8], int i)
{
      if (s[i] == '\0')
            return;

      if (!(s[i] >= '0' && s[i] <= '9'))
      {
            f(s, i + 1);
            printf("%c", s[i]);
      }
      else
      {
            printf("%c", s[i]);
            f(s, i + 1);
      }
}
```

a) atm2025 **b)** 5202mta **c)** 2025mta **d)** atm5202

132. Se consideră funcția *f* de mai jos. Care sunt parametrii *x* și *nivel* astfel încât funcția să afișeze următorul pattern:

```
*
***
*****
*******
```

```
void f(int x, int nivel)
{
	int i;
	if (x <= 0)
		return;
	f(x - 1, nivel + 1);

	for (i = 0; i < nivel; i++)
		printf(" ");

	for (i = 0; i < 2 * x - 1; i++)
		printf("*");

	printf("\n");
}
```

a) *x=3, nivel=2*
b) *x=7, nivel=0*
c) *x=4, nivel=0*
d) *x=2, nivel=3*

133. Considerând variabila globala *a* un vector de numere întregi astfel încât *a[i] = i*, primul index fiind 1 și variabila întreagă n, numărul de elemente al vectorului, ce valoare va avea apelul funcției *f(1)* pentru *n = 1024*?

```
int f(int i)
{
	int j, s = 0;
	if (i > n)
		return 0;
	for (j = 1; j <= a[i]; j++)
		s++;

	return s + f(i + 1);
}
```

a) 524 800 **b)** 523 776 **c)** 524 288 **d)** 523 700

134. Se consideră funcția recursivă definită mai jos. Care este valoarea returnată în urma apelului *f(500, 1000)*?

```
int f(int a, int b){
      if (a == b)
            return (a + b) / 2;
      else
            return 1 + f(a + 2, b - 2);
}
```

a) 750 **b)** 1100 **c)** 875 **d)** 975

135. Se consideră subprogramul de mai jos. Menționați care este valoarea returnată de funcția *g*.

```
int f(int n) {
      if (n == 0)
            return 0;
      if (n == 1)
            return 1;
      return (f(n - 1) + f(n - 2));
}
int g() {
      int i;
      for (i = 2; i < 15; i++) {
            int res = f(i);
            if (res % 2 == 0 && res % 3 == 0)
                  return res;
      }
}
```

a) 0 **b)** 144 **c)** 46368 **d)** 4181

136. Se consideră subprogramul de mai jos, unde *n* reprezintă un întreg. Menționați ce se afișează la apelul funcției print(4).

```
void print(int n){
      if (n == 0)
            return;
      print(n - 1);
      printf("%d",n);
      print(n - 1);
}
```

a) 12321412321
b) 121312141213121
c) 1122334332211
d) 12341231211212312 34

137. Se consideră subprogramul de mai jos. Menționați ce returnează apelului funcției *f*.

```
int f(int n){
      if (n <= 0)
            return 0;
      if(n%10 == (n/10)%10)
            return f(n - 1) + n;
      return f(n - 1);
}
```

a) Suma numerelor care au ultimele două cifre egale și care sunt în intervalul [*0*,*n*]
b) Suma numerelor de doua cifre din intervalul [*0*,*n*]
c) Funcția nu va returna nimic, pentru că intră într-o buclă infinită
d) Suma numerelor de două cifre din intervalul [*0*,*n*)

138. Se consideră subprogramul de mai jos. Menționați ce va returna apelul funcției *print* pentru *s = "radar"*.

```
int verificare(char s[])
{
      int n = strlen(s);
      int i;
      for (i = 0; i < n / 2; i++)
            if (s[i] != s[n - i - 1])
                  return 0;

      return 1;
}
```

a) -2 **b)** -1 **c)** 0 **d)** 1

139. Ce afișează programul de mai jos?

```
int f(int v[], int n)
 {
      if (n == 1)
      return v[0];
      int m = f(v, n - 1);
      return (v[n - 1] > m) ? v[n - 1] : m;
}
void main()
{
      int v[] = {3, 7, 2, 8, 6, 4};
      f(v, 6);
}
```

a) 3 **b)** 7 **c)** 2 **d)** 8

140. Ce se va afișa pe ecran în urma executării următorului program?

```
int f(int n)
{
	if (n >=0 && n < 10)
		return n;

	int aux = n;
	int nr_cifre = 1;

	while (aux>=10)
	{
		aux /= 10;
		nr_cifre *= 10;
	}

	return f(n / 10) + (n % 10)*nr_cifre;

}
void main()
{
	int numar = f(123456);
	printf("%d", numar);
}
```

a) 654321 **b)** 654312 **c)** 123456 **d)** 0

141. Precizați ce se va afișa în urma apelului funcției *func(874576329)*.

```
int func(int n)
{
	if (n >= -10)
		printf("%d", n % 10);

	if (n < 0)
		return 0;

	n++;
	if ((++n) % 2 == 0)
		n /= 100;
	else
		n /= 10;

	return func(n - 7);
}
```

a) 9690791-7 **b)** 9680871-7 **c)** 968971-7 **d)** 9658087-7

142. Dată fiind funcția *rec*, ce returnează apelul *rec(287349176)*?

```
int rec(int x){
      if (x == 0)
            return 0;
      if (!(x % 2))
            return 1 + rec(x / 10);
      return 1000 + rec(x / 10);
}
```

a) 6003 **b)** 4005 **c)** 3006 **d)** 5004

143. Fie funcția *f* de mai jos. Ce valoare returnează de apelul *f(9875)*.

```
int f(int n){
      if (n < 10)
            return n;
      int s=0;
      while (n > 0)
      {
            s += n % 10;
            n /= 10;
      }
      return f(s);
}
```

a) 11 **b)** 5 **c)** 29 **d)** 2

144. Ce putem afirma despre programul de mai jos?

```
int function(int x, int b) {
      if (x == 0)
            return b;

      if (x % 10 % 3 == 0)
            b = b * 10 + x % 10;
      return function(x / 10, b);
}
void main(){
      int x = function(5738926, 0);
      if (x % 3 == 0)
            printf("YES");
}
```

a) calculează inversul unui număr
b) afișează YES
c) nu afișează nimic
d) calculează suma cifrelor divizibile cu 3

145. Fiind dată funcția *f* de mai jos, ce va returna apelul *f(3,5)* ?

```
int f(int x, int y) {
      if (x % 2 == 1)
            x++;
      else
            x = x / 2;
      if (y > x)
            y = y - x + 1;
      else
            y++;
      if (x == 1)
            return y;
      else
            return f(x, y + 1) + f(y, x);
}
```

a) 7 **b)** 4 **c)** 9 **d)** 3

146. Se consideră subprogramul de mai jos, unde *n, m și k* reprezintă numere întregi. Menționați care este rezultatul apelului funcției *F(4, 2, 3)*:

```
int F(int n, int m, int k) {
      if (n == 0)
            return m;
      else
            return (n % k) * m + F(n - 1, m + 1, k);
}
```

a) 21 **b)** 20 **c)** 26 **d)** 0

147. Se consideră subprogramul de mai jos. Cum trebuie apelat subprogramul *fun*, astfel încât să se afișeze pe ecran întreg șirul *text* în ordine inversă?

```
void fun(char str[], int k) {
      if (str[k] == 0)
            return;
      else {
            k = k + 1;
            fun(str, k);
            printf("%c", str[k]);
      }
}
```

a) *funString(text,-1)*
b) *funString(text, 0)*
c) *funString(text, 2)*
d) *funString(text,-2)*

148. Se consideră subprogramul de mai jos. Menționați ce va afișa apelul *func* pentru *n = 32*.

```
int func(int n){
      if (n % 3 == 2)
            return 1 + func(n - 1);
      if (n % 3 == 1)
            return 2 + func(n / 3);
      return 5;
}
```

a) 7 **b)** 9 **c)** 10 **d)** 12

149. Cunoscând subprogramul *g*, definit mai jos, ce valoare se va obține în urma apelului *g(6, 4)*?

```
int g(int x, int y) {
      if (x == 0 || y == 0)
            return x + y;
      return g(y - 1, x - 2) + x - y;
}
```

a) 5 **b)** 7 **c)** 4 **d)** 9

150. Considerând funcția *f* de mai jos, precizați ce valoare va returna apelul *f(9112406)*?

```
int f(int x) {
      if (x == 0)
            return x;
      if (x % 2 == 0)
            return 1 + f(x / 10);
      else
            return f(x / 10);
}
```

a) 3 **b)** 2 **c)** 4 **d)** 0

151. Precizați ce valoare întoarce apelul funcției *f(2518179,24)* pentru funcția *f* definită mai jos:

```
int f(int n, int value){
      if (n < 10)
            return value;
      int nr = ((n / 10) % 10) * 10 + n % 10;
      if (nr > value)
            value = nr;
      f(n / 10, value);
}
```

a) 89 **b)** 25 **c)** 81 **d)** 79

152. Precizați valoarea numărului întreg *n* pentru care apelul *f(n)* al funcției *f* definită mai jos returnează valoarea *10*.

```
int f(int n)
{
      if (n < 10) return n;
      return n % 10 + f(n / 10);
}
```

a) 456780 **b)** 1040212 **c)** 0 **d)** 10

153. Cu ce expresie trebuie înlocuită zona marcată cu . . . . . , pentru ca în urma execuției programului de mai jos să se afișeze valoarea 8?

```
int fun(char v[100], int i) {
      if (v[i + 1] == 0)
            return 1;
      else
            . . . . . .;
}
int main() {
      char v[100] = "ACADEMIA";
      printf("%d\n", fun(v, 0));
      return 0;
}
```

a) *return fun(v,i)+1*
b) *return fun(v,i+1)+1*
c) *fun(v,i+1)*
d) *fun(v,i)+1*

154. Funcția *fun* își propune afișarea șirului de caractere *v* în ordine inversă, însă conține o greșeală. Indicați linia de cod la care este greșeala și instrucțiunea corectă pentru ca subprogramul să îndeplinească corect sarcina.

```
1:  void fun(char v[], int n)
{
2:    if (!v[n])
3:          return;
4:    fun(v, n);
5:    printf("%c ", v[n]);
6: }
```

a) linia 4, `fun(v,n+1)`
b) linia 2, `if(v[n])`
c) linia 2, `if(v[n]==0)`
d) linia 4, `fun(v,n-1)`

155. Se consideră subprogramul de mai jos. Deduceți rolul acestui subprogram și apoi completați zona marcată cu . . . . . , astfel încât subprogramul să îndeplinească corect sarcina pentru care a fost realizat.

```
int fun(char str[], char c, int i) {
      . . . . . .
            return -1;
      if (str[i] == c)
            return i;
      return fun(str, c, i + 1);
}
```

a) *if(!str[i])*
b) *if(str[i]!=0)*
c) nu mai trebuie adăugat nimic în program
d) *if(str[i]!=c)*

BACKTRACKING ȘI PROBLEME DE GENERARE

156. Referitor la tehnica backtracking, câte dintre următoarele afirmații sunt corecte?
I – Poate fi utilizat pentru a genera toate soluțiile unei probleme
II – Poate fi utilizat eficient la sortarea unui vector
III – Nu poate fi implementat fără recursivitate
IV – Căutarea binară este un exemplu de algoritm de tip backtracking

a) 0 **b)** 1 **c)** 2 **d)** 4

157. Se dorește generarea, în ordine strict crescătoare, a tuturor numerelor din intervalul $[100, 400]$ care sunt prime permutabile (numere pentru care toate permutările cifrelor lor sunt prime). Primele *4* soluții generate sunt: *113*, *131*, *199* și *311*.
Care este următoarea soluție generată de algoritm?

a) 317 **b)** 371 **c)** 337 **d)** 373

158. Se consideră toate numerele de cel mult *3* cifre care sunt generate, în ordine, folosind cifrele din mulțimea {*0, 1, 2, 3, 4, 5*} cu mențiunea că suma cifrelor este egală cu *9*. Primele trei sunt: *135*, *144*, *153*. Care sunt al *7*-lea număr și ultimul număr generat?

a) 243, 540 **b)** 315, 531 **c)** 252, 540 **d)** 333, 522

159. Maria are *5* cartonașe cu numerele *1*, *3*, *3*, *6*, *6* și dorește să genereze într-o ordine dată toate numerele diferite de *5* cifre. Dacă primele *4* numere generate de ea sunt *13366*, *13636*, *13663*, *16336*, precizați care este al *12*-lea număr generat de Maria.

a) 16363 **b)** 31663 **c)** 33661 **d)** 36631

160. Se generează secvențe de câte *4* litere distincte din primele 6 litere ale alfabetului. Secvențele respectă următoarele condiționări:
- oricare două litere consecutive nu pot fi pe poziții alăturate.
- toate secvențele încep cu litera *a*.

Câte astfel de secvențe se pot genera?

a) 22 **b)** 23 **c)** 14 **d)** 15

161. Fie *s* un șir de caractere. Precizați efectul apelului *f2(s,0,n)*, unde *n=strlen(s)-1*.

```
void f(char s[], int i, int j)
{
	char temp = s[i];
	s[i] = s[j];
	s[j] = temp;
}

void f2(char s[], int i, int j)
{
	if (i == j)
	{
		printf("%s\n", s);
	}
	else
	{
		int k;
		for (k = i; k <= j; k++)
		{
			f(s, i, k);
			f2(s, i + 1, j);
			f(s, i, k);
		}
	}
}
```

a) Generează și afișează toate subșirurile șirului de caractere *s*.
b) Verifică dacă șirul de caractere *s* este un palindrom
c) Generează și afișează toate permutările șirului de caractere *s*
d) Verifică dacă șirul de caractere *s* conține un anumit subșir

GRAFURI NEORIENTATE

162. Un graf neorientat *G* are *n=20* noduri și *m=25* muchii. Care este suma elementelor din matricea de adiacență?

a) $20 * 20$ **b)** $25 * 2$ **c)** $20 * 25$ **d)** $25 * 25$

163. Un graf *G* are *20* de noduri, *30* de muchii și este reprezentat printr-o matrice de adiacență. Care este dimensiunea acestei matrice?

a) *20x20* **b)** *20x2* **c)** *20x30* **d)** *30x30*

164. Se dă un graf neorientat G cu 5 noduri V = *{0, 1, 2, 3, 4}* și lista de adiacență M = *{(0, 1), (1, 2), (2, 3), (3, 4)}*. Câte din nodurile lui G au cel puțin 2 vecini?

a) 0 **b)** 5 **c)** 3 **d)** 1

165. Fie graful neorientat *G* cu 5 noduri, *V* = {0, 1, 2, 3, 4} și lista de adiacență *M = {(0, 1), (1, 2), (2, 3), (3, 4)}*. Care este gradul cumulat al tuturor nodurilor din graful G?

a) 6 **b)** 4 **c)** 8 **d)** 10

166. Se consideră matricea de adiacență *A*, definită mai jos pentru un graf neorientat cu 5 noduri: *V* = {0, 1, 2, 3, 4}. Câte vârfuri izolate are graful?

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

a) 0 **b)** 1 **c)** 3 **d)** 4

167. Fie un graf *G* cu *V* noduri. Graful complementar *C* al grafului *G* este un graf format din aceleași noduri *V*, însă cu muchii astfel încât doar nodurile neadiacente din *G* sunt adiacente în *C*. Considerăm că *G* este un graf neorientat cu *5* noduri, în care nodurile *1* și *4*, respectiv *3* și *5*, sunt adiacente. Care este numărul de muchii al grafului complementar?

a) 10 **b)** 4 **c)** 8 **d)** 6

168. Fie un graf neorientat, pentru care numerotarea nodurilor începe de la 0, reprezentat prin următoarea matrice de adiacență:

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

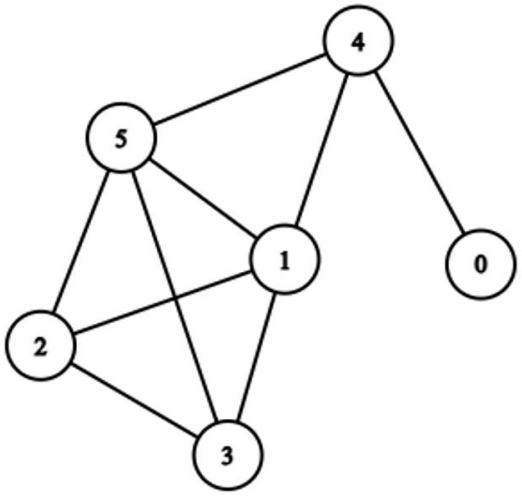
Care dintre următoarele afirmații este *falsă*:

a) Graful este conex
b) Nodul 1 are gradul 2
c) Graful conține un ciclu
d) Matricea de adiacență este simetrică

169. Se dă un graf neorientat cu nodurile *1, 2, 3, 4, 5, 6* și muchiile *(1,2), (1,3), (3,4), (4,5).* Selectați afirmația falsă:

a) Nodul 6 este izolat
b) Este nevoie de încă minim 2 muchii pentru a fi graf hamiltonian
c) Este nevoie de încă minim 2 muchii pentru a fi graf eulerian
d) Graful are 2 componente conexe

170. Afirmația adevărată cu privire la următorul graf este:



a) Graful nu conține cicluri, deoarece este neorientat
b) Gradul nodului 1 este 5
c) Nodul 0 este un nod izolat
d) Gradul nodului 2 este 3

171. Având graful *G* definit prin următoarele muchii:
G={[1,2],[1,3],[1,4],[1,5],[2,3],[2,4],[2,7],[3,4],[3,6],[4,6],[4,8],[5,7],[5,8],[6,7],[6,8],[7,8]}
Care dintre următoarele afirmații este corectă în legătură cu graful *G*?

a) Graful este conex, hamiltonian și nu este eulerian
b) Graful este conex, hamiltonian și eulerian
c) Graful nu este conex și nu este hamiltonian
d) Graful este conex, nu este eulerian și nu este hamiltonian

172. Câte grafuri neorientate distincte de 5 noduri se pot construi, dacă un nod oarecare este mereu izolat?

a) 0
b) exact $5 * 2^6$
c) mai multe decât $5 * 2^6$
d) mai puține decât $5 * 2^6$

173. Fie un graf neorientat cu *N=7* noduri. Selectați varianta care poate reprezenta mulțimea gradelor nodurilor acestui graf, având în vedere că graful este conex.
a) {1, 2, 3, 1, 0, 1, 2}
b) {2, 5, 1, 4, 3, 1, 6}
c) {1, 1, 1, 1, 1, 2, 1}
d) {3, 1, 4, 1, 1, 2, 1}

174. Care dintre următoarele liste de adiacență generează un graf neorientat cu *3* componente conexe?

a)
1: 2
2: 1, 3
3: 2
4:
5: 6
6: 5

b)
1: 2, 3, 4
2: 1, 3
3: 1, 2, 4
4: 1,3
5: 6
6: 5

c)
1: 5
2: 3, 4
3: 2
4: 2
5: 1, 6
6: 5

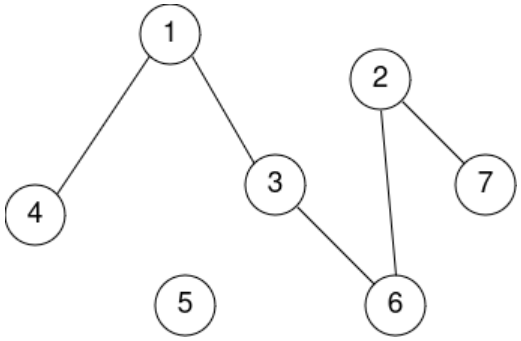
d)
1:
2: 3, 6
3: 2
4:
5:
6: 2

175. Câte noduri terminale are graful descris de următoarea matrice de adiacență?

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

a) 0 **b)** 2 **c)** 3 **d)** 1

176. Fie graful din figura de mai jos. Care este numărul minim de muchii ce trebuie adăugate astfel încât subgraful alcătuit din vârfurile *1, 3, 4, 7* să fie un graf complet?



a) 4 **b)** 0 **c)** 15 **d)** 2

177. Care este numărul de muchii al grafului dat prin matricea de adiacență de mai jos?

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

a) 7 **b)** 8 **c)** 9 **d)** 6

178. Se consideră că funcția *getdegree(v)* întoarce pentru graful *G*, gradul nodului *v*. Cunoscând că *G* are *4* noduri și că în urma executării codului de mai jos, rezultatul este: *2, 2, 1, 3*, care este matricea de adiacență a lui *G* ?

```
void main(){
      int i;
      for (i = 0; i < 4; i++)
            printf("%d", getdegree(i));
}
```

a) $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$ **c)** $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$

b) $\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$ **d)** $\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$

179. Alegeți matricea de adiacență pentru care graful asociat conține *3* componente conexe.

a) $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$ **b)** $\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

c) $\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$ **d)** $\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

180. Se consideră subprogramul de mai jos, unde *G* este un graf neorientat cu 5 noduri reprezentat prin matricea de adiacență. Programul de mai jos afișează valoarea 10. Ce poate fi afirmat cu certitudine despre graful *G*?

```
void print(int G[5][5]){
      int i, j, suma=0;
      for (i = 0; i < 5; i++)
            for (j = 0; j < 5; j++)
                  suma += G[i][j];
      printf("%d", suma);
}
```

a) Graful are exact 5 muchii
b) Graful are o singură componentă conexă
c) Graful conține un lanț elementar
d) Graful conține un lanț neelementar de lungime 5

181. Se consideră graful neorientat *G* reprezentat prin listele de adiacență de mai jos. Este graful *G* conex?

```
0: 1, 2
1: 0, 3
2: 0, 4
3: 1, 4
4: 2, 3
```

a) Da
b) Nu
c) Este complet, deci conex
d) Este aciclic, deci conex

182. Se consideră un graf neorientat reprezentat prin matricea de adiacență de mai jos. Care dintre următoarele sunt gradele nodurilor din acest graf?

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

a) 2, 3, 3, 2, 2 **b)** 3, 2, 3, 3, 2 **c)** 2, 3, 3, 3, 2 **d)** 2, 3, 4, 2, 2

183. Câte grafuri parțiale distincte suportă un graf neorientat cu N noduri?

a) N **b)** $2^{\frac{N(N-1)}{2}}$ **c)** $2^{\frac{N(N+1)}{2}}$ **d)** $2^{\frac{N}{2}}$

184. Se consideră un graf neorientat, complet, cu *10* noduri. Câte subgrafuri distincte cu 3 muchii se pot forma?

a) 240 **b)** 120 **c)** 1024 **d)** 14190

185. Se consideră un graf neorientat cu *20* noduri. Câte subgrafuri distincte cu maximum *2* muchii se pot forma?

a) 18146 **b)** 18145 **c)** 118200 **d)** 20112

186. Se dă un graf neorientat definit de nodurile $V = \{ 1, 2, 3, ... 1000 \}$ și setul de muchii $E = \{ (x, y), y = 2x + 3, x \in V$ și $y \in V \}$. Care propoziție este adevărată?

a) Graful are 498 de muchii
b) Graful are cel puțin 500 de muchii
c) Graful este conex
d) Gradul fiecărui nod este 2

187. Afirmația adevărată cu privire la următorul graf dat prin matricea sa de incidență este:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

a) Graful nu conține cicluri
b) Gradul nodului 0 este 3
c) Nodurile 0 și 4 au gradul 6
d) Nodurile grafului au același grad

ARBORI

188. Un arbore binar este un arbore pentru care fiecare nod poate avea *0*, *1* sau *2* fii. Fie un astfel de arbore binar cu *31* de noduri. Care este numărul minim de niveluri pe care îl poate avea acesta?

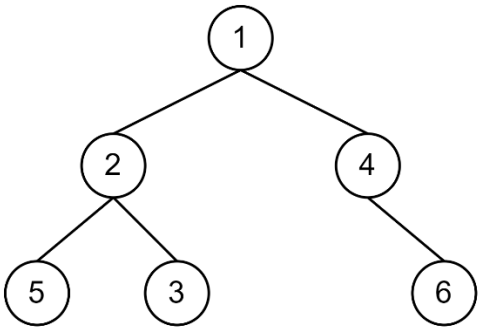
a) 4 **b)** 5 **c)** 6 **d)** 7

189. Câte frunze are arborele descris de următoarea matrice de adiacență?

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

a) 0 **b)** 2 **c)** 3 **d)** 1

190. Alegeți afirmația adevărată cu privire la arborele de mai jos.



a) parcurgerea inordine a arborelui este: 5 3 6 2 4 1
b) parcurgerea preordine a arborelui este: 1 2 5 4 3 6
c) parcurgerea preordine a arborelui este: 5 3 2 6 4 1
d) parcurgerea inordine a arborelui este: 5 2 3 1 4 6

191. Se consideră un arbore cu *46* de noduri, iar fiecare nod care nu este frunză are cel puțin *5* fii. Care este înălțimea maximă a unui arbore care să respecte condițiile menționate?

a) 9 **b)** 10 **c)** 3 **d)** 4

192. Se consideră un vectorul de tați al unui arbore cu *9* noduri definit astfel: *[6 3 0 6 7 2 3 6 2]*. Indicați numărul de frunze al acestui arbore.

a) 3 **b)** 5 **c)** 4 **d)** 7

193. Se consideră vectorul de părinți $T = [0,\ 1,\ 2,\ 2,\ 2,\ 5,\ 1,\ 7]$, pentru a reprezenta arborele A. Ce alt vector de părinți poate fi format cu nodurile și muchiile arborelui A, pentru a obține un arbore cu adâncimea maximă posibilă?

a) $[2, 5, 2, 2, 6, 0, 1, 7]$
b) $[2, 4, 2, 0, 2, 5, 1, 7]$
c) $[2, 0, 2, 2, 2, 5, 1, 7]$
d) $[7, 1, 2, 2, 2, 5, 0, 7]$

194. Un arbore cu 12 noduri, numerotate de la 1 la 12, are drept rădăcină nodul 7 și muchiile [7,3], [7,8], [3,1], [3,4], [8,9], [8,10], [1,2], [1,5], [4,6], [4,11], [10,12]. Care este numărul de descendenți direcți pentru nodul 4 și care este înălțimea subarboreului cu rădăcina în nodul 3?

a) 3 descendenți, *h=3*
b) 2 descendenți, *h=3*
c) 3 descendenți, *h=2*
d) 2 descendenți, *h=2*

195. Se consideră matricea adiacență de mai jos, a unui arbore binar cu rădăcina nodul *3*. Considerând că nodurile sunt notate de la *1* la *6*, câte noduri copil are nodul *1*?

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

a) 0 **b)** 1 **c)** 2 **d)** 3

196. Fie un arbore binar cu *63* de noduri. Un arbore se numește arbore binar dacă fiecare nod are maximum *2* fii. Care este numărul maxim de frunze pe care îl poate avea arborele menționat?

a) 37 **b)** 15 **c)** 23 **d)** 32

197. Se consideră un arbore cu 10 noduri, numerotate de la 1 la 10 având vectorul de tați următor *[5, 3, 7, 7, 0, 5, 5, 3, 5, 4]*. Care este nodul de grad maxim din arbore?

a) 3 **b)** 7 **c)** 5 **d)** 4

PROBLEME DE NIVEL UȘOR-MEDIU. SOLUȚII

1. Condiția *a>b && b<c* nu este respectată, prin urmare se execută ramura *else*. Apoi, întrucât *b>c*, se afișează *3*.

Răspunsul corect este **a)**.

2. În expresia *x=y/z+2* se calculează împărțirea a două variabile de tip *int*, adică 10/4, rezultatul fiind 2.

Răspunsul corect este **b)**.

3. După prima expresie de incrementare a variabilei *a*, aceasta va avea valoarea *6*. La a doua expresie, *b* devine 6 iar *a=a+6=12*.

Răspunsul corect este **d)**.

4. Pentru că *a* este diferit de *0*, atunci expresia booleană *!a* este evaluată ca fiind *true*. Atunci când interpretăm ca întreg valoarea booleană *true*, aceasta este convertită la *1*.
Expresia *!!a* este evaluată la *false* și convertită ulterior la *0*.

Răspunsul corect este **a)**.

5. Programul extrage cifrele numărului de la dreapta la stânga folosind instrucțiunea *n%10*. Apoi elimină cifra afișată folosind instrucțiunea *n/=10*. În final, programul afișează cifrele numărului în ordine inversă.

Răspunsul corect este **a)**.

6. Presupunem un *a* pentru care *2025>a*.
La prima iterație, *x=2024*, *x%4 = 0*, printează Admitere_
La a doua iterație, *x=2023*, *x%4 = 3*, printează ATM (afișarea a fost realizată)
A treia iterație nu mai trebuie sa fie permisă. Așadar, *a=2023*.

Răspunsul corect este **c)**.

7. Instrucțiunea `m = (m < d) ? d : m;` va stoca în variabila m maximul dintre m și d. Astfel, în urma parcurgerii celor 10 elemente, m va fi egal cu cel mai mare element introdus.

Răspunsul corect este **b)**.

8. Expresia este evaluată la 1 (este *adevărată*) dacă și numai dacă:
- condiția $a < b$ este *adevărată* și condiția $c < d$ este *adevărată*, adică $a < b$ și $c < d$, sau
- condiția $a < b$ este *falsă* și condiția $c < d$ este *falsă*, adică $a >= b$ și $c >= d$

Singurul răspuns din cele furnizate, care este identic cu una dintre cele două variante menționate mai sus este răspunsul de la varianta *d*.

Răspunsul corect este **d)**.

9. Atunci când *i* este par, condiția din bucla *while* va fi mereu diferită de 0, prin urmare, se va genera o buclă infinită.

Răspunsul corect este **a)**.

10. Operatorul împărțire aplicat între doi operanzi întregi realizează împărțirea fără rest. Așadar, analizăm subexpresiile astfel:
Primul calcul, *++x*, face ca *x* să devină *2026*.

$$y = 2026 + 2026/10 + 2026\%\,10 = 2026 + 202 + 6 = 2234$$

Răspunsul corect este **c)**.

11. Subprogramul verifica daca *x* este un număr prim.

Răspunsul corect este **d)**.

12. Programul returnează cel mai mare număr prim mai mic decât o valoare dată. Pentru *n=100*, numerele prime mai mici decât *100* sunt: *97, 89, ... 7, 5, 3, 2*.

Răspunsul corect este **a)**.

13. Algoritmul lui Euclid calculează cel mai mare divizor comun (*cmmdc*) iterativ. Resturile succesive sunt*:*
48 % 18 = 12,
18 % 12 = 6,
12 % 6 = 0. deci, cmmdc este 6.

Răspunsul corect este **a)**.

14. Bucla *for* parcurge toate numerele de la *1* la *10*. Condiția *i % 2 == 0* verifică dacă un număr este par. Numerele pare din intervalul [*1, 10*] sunt: *2, 4, 6, 8, 10*. Suma acestora este: *2 + 4 + 6 + 8 + 10 = 30.*

Răspunsul corect este **b)**.

15. Funcția *print* afișează numerele prime din vector. *10341* este divizibil cu *3*, așadar nu este afișat.

Răspunsul corect este **d)**.

16. Funcția *count* numără câte cifre impare sunt în numărul primit ca parametru. Expresia *(value % 10) % 2 != 0* este echivalentă cu *value % 2 != 0*.

Răspunsul corect este **d)**.

17. Programul iterează prin toate valorile din intervalul *[0,99]*, verifică care dintre aceste valori sunt divizibile cu *2* și al căror rest în urma împărțirii cu 3 are valoare 1. Variabila *s* va conține suma acestor numere.

Răspunsul corect este **c)**.

18. Se verifică fiecare dintre expresii, respectând ordinea efectuării operațiilor (adunare și scădere după înmulțire și împărțire).
Observație 1: $x/10^n$ *va elimina ultimele n cifre din x.*
Observație 2: $x\%10^n$ *va selecta ultimele n cifre din x.*
Răspunsul corect este **d)**.

19. Subprogramul calculează produsul divizorilor primi. Bucla while face împărțirea lui x la unul dintre divizorii săi, până când x devine indivizibil cu i. Operația este necesară pentru că se dorește obținerea produsul divizorilor distincți. (ex. pentru $x = 24$, 2 se va lua în calcul o singură dată).
Răspunsul corect este **d)**.

20. Ordinea efectuării operațiilor ține cont de prioritatea parantezelor. Variabila *a* are valoarea *4*, iar variabila *b* are valoarea *6*.
Răspunsul corect este **c)**.

21. Variabile x, y, z sunt globale. În funcția f, variabilele x, z sunt locale astfel încât cele locale sunt prioritare, respectiv, în funcția *main*, există variabila locală x. De asemenea, operația de preincrementare (++) are precedență mai mare decât operatorul de atribuire, respectiv operația de postdecrementare se va realiza ca ultimă operație din expresie.
Răspunsul corect este **a)**.

22. Numele de variabile trebuie să înceapă cu o literă mică, literă mare sau bară jos (underscore); poate conține litere, cifre sau underscore; nu pot fi folosite cuvinte rezervate ale limbajul C.
Răspunsul corect este **a)**.

23. Analizăm expresia pe subexpresii, astfel:
(a && !b) = 1 se respectă dacă și numai dacă *a = 1* și *b = 0*
(a && (!b || !a)) = 1 se respectă dacă și numai dacă *a = 1* și (*b= 0* sau *a = 0*)
Pentru cele două subexpresii se face operația de și logic (intersecție), prin urmare expresia completă este respectată dacă și numai dacă *a = 1* și *b = 0.*
Răspunsul corect este **b)**.

24. Se execută mai întâi *++b, b* devine 14. Apoi *c = 14*, apoi *b = 14*, apoi *a = a+ b = 10+14 = 24*. Linia de cod comentată */* a++; */* este ignorată. Ultima linie de cod face ca *a = a + 2 = 26*.
Răspunsul corect este **d)**.

25. Programul numără câte cifre din numărul întreg *b* sunt strict mai mari decât cifra zecilor din numărul întreg *a*.

Răspunsul corect este **c)**.

26. Bucla *for* realizează parcurgerea vectorului iar la fiecare iterație este obținută suma parțială a vectorului; *if* va selecta doar elementele care sunt mai mari decât suma parțială a vectorului până la poziția curentă.

Răspunsul corect este **b)**.

27. Funcția verifică dacă numărul n este egal cu suma divizorilor săi. Dintre numerele pozitive mai mici decât 10, singurul care respectă această condiție este 6.

Răspunsul corect este **c)**.

28. Subrutina calculează suma numerelor impare, divizibile cu *n* și mai mici sau egale cu *m*.
Se observă că apelul *prog(5,3)=3*, ceea ce invalidează răspunsul a).
Se observă că apelul *prog(5,1)=9*, ceea ce invalidează răspunsurile b) și c).

Răspunsul corect este **d)**.

29. Instrucțiunea *switch* va verifica restul împărțirii la *3* și pentru fiecare caz atribuie o valoare variabilei x. Verificând valorile din cele *4* variante de răspuns, determinăm că *x=22* va rezulta *y=2* iar *x=24* va rezulta *y=4*.

Răspunsul corect este **a)**.

30. La o analiză atentă a funcției print, se observă că *var* este construit pe baza cifrelor lui *n,* după următoarea regulă: dacă o cifra din *n* este pară, cifra corespunzătoare din *var* va fi *0,* altfel va fi *1*. Programul poate fi analizat pas cu pas, având 9 iterații.

Răspunsul corect este **d)**.

31. La prima iterație, *i=0:*
`printf("%d ", i++);` afișează *0* și apoi i devine *1*.
`printf("%d ", --i);` decrementează *i* înapoi la *0* și afișează *0*.
Se realizează incrementarea pentru *i*, specifică buclei *for*.
Bucla se oprește deoarece *i* nu mai satisface `i%2==0`, *i* având valoarea *1*.

Răspunsul corect este **b)**.

32. Se parcurge fiecare iterație a buclei while, până când *x* devine mai mic decât *a*. Cele două valori se modifică astfel:
- dacă *a* este divizor al lui *x,* se scade *a* din *x* și *a* se multiplică cu *10*.
- dacă *a* nu este divizor al lui *x*, *x* se împarte la *a* și *a* se incrementează.

Răspunsul corect este **c)**.

33. Operațiile expresiei se execută astfel:

$a \,\%\, (a \,+\, 1) \,/\, (a \,\%\, 4 \,+\, 1) \,+\, a/10/10;$
$a \,\%\, 16 \,/\, (a \,\%\, 4 \,+\, 1) \,+\, a/10/10;$
$a \,\%\, 16 \,/\, (3 \,+\, 1) \,+\, a/10/10;$
$a \,\%\, 16 \,/\, 4 \,+\, a/10/10;$
$a \,\%\, 4 \,+\, a/10/10;$
$a \,\%\, 4 \,+\, 1/10;$
$a \,\%\, 4 \,+\, 0;$
$3 \,+\, 0\,;$

Răspunsul corect este **a).**

34. Analizăm fiecare instrucțiune de atribuire în parte:

$$x \mathrel{+}= z \,\%\, 3 \Leftrightarrow x = 1 + 2\%3 \Rightarrow x \; devine \; 3$$
$$y = x \,/\, 2 \Leftrightarrow y = 3 \,/\, 2 \Rightarrow y \; devine \; 1$$
$$z \mathrel{+}= (x + y) \,\%\, 2 + z \Leftrightarrow z = 2 + (3 + 1) \,\%2 + 2 \Rightarrow z \; devine \; 4$$

Răspunsul corect este **a).**

35. Funcția f realizează verificarea dacă un număr este prim. Suplimentar, *f* conține și condiția de apartenență la un interval. În cazul de față, numărul *n* ar trebui să aparțină intervalului $[a, b)$ pentru ca *f* să returneze *1*.

Răspunsul corect este **d).**

36. Bucla *for* parcurge pe rând toate numerele de la a până la predecesorul lui b (condiția este $i < b$), deci numerele afișate vor aparține intervalului $[a, b)$. În interiorul primei bucle *for* se verifică dacă numărul curent este prim. Ultimul *if* verifică dacă ultima și penultima cifră sunt egale. Numerele care respectă cele două condiții sunt afișate.

Răspunsul corect este **b).**

37. Bucla *while* va înjumătăți valoarea variabilei a de $b/2$ ori, unde b este restul împărțirii lui a la *100*.

Răspunsul corect este **a).**

38. Condiția verifică dacă există un divizor comun pentru a și b în intervalul $[2;\ min(a, b)]$. În cazul identificării unui divizor, va atribui variabilei condiție valoarea *1*, marcând astfel că există cel puțin un divizor comun al celor două numere.

Răspunsul corect este **d).**

39. Vectorul v[5] se inițializează cu {1 0 0 0 0}. După parcurgerea vectorului și executarea instrucțiunilor din blocul switch, vectorul va conține valorile 2 1 1 1 1, care vor fi ulterior afișate.

Răspunsul corect este **b).**

40. Analizăm fiecare iterație a buclei *while*, astfel:
iteratie 1: `x=54, y=12, x%y=6, x/y=4`
iteratie 2: `x=56, y=11, x%y=1, x/y=5`
iteratie 3: `x=58, y=10, x%y=8, x/y=5`
iteratie 4: `x=60, y= 9, x%y=6, x/y=6`
Răspunsul corect este **b)**.

41. Varianta de răspuns *a)* verifică dacă *x* este mai mare sau egal decât *1* și mai mic strict decât *3*.
Răspunsul corect este **a)**.

42. Funcția *f* returnează valoarea dublată a parametrului *x* însă nu modifică parametrul *x*. Instrucțiunea `y = f(++x)` modifică valoarea lui *x* la *2* înainte de apelul funcției *f* și apoi apelează funcția *f* pentru *x = 2*.
Răspunsul corect este **d)**.

43. Funcția calculează numărul de divizori primi diferiți ai parametrului *a*. Se descompune numărul *1091740* în divizori și se obțin *5* divizori primi diferiți *{2, 5, 13, 17, 19}*.
Răspunsul corect este **b)**.

44. Un element se află pe poziție impara dacă respectă condiția *i%2*==0, și este impar dacă respectă condiția *a[i] % 2 == 1*.
Răspunsul corect este **d)**.

45. Tabloul este bidimensional (matrice). Din definiția tabloului rezultă imediat că acesta conține 6 valori. Ceea ce înseamnă că acest tablou are 3 linii și 2 coloane (indexate de la 0). Altfel spus, matricea poate fi scrisă și astfel: *T[3][2] = { {1, 2}, {3, 4}, {5, 6} }*.
T[1][1] este al doilea element de pe a doua linie a tabloului.
Răspunsul corect este **c)**.

46. Programul parcurge elementele de pe diagonala principală și pe cele de pe diagonala secundară, în ordine inversă. Afișează o valoare de pe diagonala principală urmată de o valoare de pe diagonala secundară.
Răspunsul corect este **d)**.

47. Prețul obiectelor de pe poziția i este $nr\_buc[i] * preturi[i]$. Suma totală se calculează iterând de la *0* până la $n - 1$, și însumând valorile obținute pentru fiecare produs.
Răspunsul corect este **d)**.

48. Analizând răspunsurile putem observa că prima linie și prima coloană fiind *0*, doar răspunsurile cu $i * j$ au sens. Dintre cele 2 variante, doar cea cu $(i * j)\%N$ poate fi corectă.
Considerăm cazul $i = 4$ și $j = 4 \Rightarrow a[4][4] = 1$.

Răspunsul corect este **a)**.

49. În matricea *m* sunt *10* elemente pe diagonala secundară. Având în vedere instrucțiunea `m[i][j] = (2 * j - i) % 10` se poate observa că proprietatea de paritate este dată de valoarea *i*. În calcularea celor 10 elemente de pe diagonala secundară, *i* va fi de *5* ori par și de *5* ori impar.

Răspunsul corect este **c)**.

50. Secvența populează o matrice de nxn. Condiția din *if* este adevărată cât timp *i* și *j* sunt diferite, adică pentru elementele care nu sunt pe diagonala principală. Astfel, matricea va conține pe diagonala principală suma indicilor $(i + j)$, iar celelalte elemente, vor fi egale cu $indicele\ coloanei + 1$.

Răspunsul corect este **a)**.

51. Observăm că programul parcurge vectorul și determină minimul înregistrat în variabila *diff*. Astfel ceea ce trebuie să înlocuiască linia punctată este determinarea diferenței absolute dintre $v[i]$ și $v[i + 1]$. Pentru a realiza acest lucru, se poate calcula diferența, iar dacă aceasta este număr negativ, i se atribuie opusul său. $(diff = -diff)$.

Răspunsul corect este **a)**.

52. Programul calculează suma elementelor de pe marginea matricei *mat*, adică elementele de pe prima linie, ultima linie, prima coloană și ultima coloană.

Răspunsul corect este **a)**.

53. Parcurgerea elementelor de pe diagonala secundară și de deasupra acesteia implică respectarea condiției $i + j \leq n - 1$. Observăm că pentru a atribui valori pe toate elementele primei linii, doar variantele *a* și *b* pot face completarea corectă. Apoi, la iteratia *2* observăm că varianta *b* nu mai poate seta elemental $matrix[1][0]$.

Răspunsul corect este **a)**.

54. Formula de extragere a elementelor de pe diagonala secundară a unei matrice este $mat[i][(n - 1) - i]$, cu $0 <= i < n$, unde *n* este numărul de linii, respectiv, coloane dintr-o matrice pătratică. Diagonala secundară conține elementele 3,5,7. Suma acestora este 15, prin urmare, programul va calcula suma elementelor de pe diagonala secundară și va afișa 15.

Răspunsul corect este **b)**.

55. Funcția *calculeaza* numără de câte ori apare elementul *x=3* în vector.
Răspunsul corect este **c)**.

56. Programul ignoră numerele pare citite și le salvează în vector doar pe cele impare. Ulterior, afișează elementele vectorului.
Răspunsul corect este **d)**.

57. Având în vedere faptul că funcția este apelată cu dimensiunile 4, 4, acestea fiind egale cu $n-1$, parcurgerea matricei va exclude ultima linie și ultima coloană. Dintre elementele parcurse, vor fi afișate doar valorile care se regăsesc pe linii pare și pe coloane impare.
Răspunsul corect este **d)**.

58. Matricea calculată are elementele:

$$\begin{pmatrix} 4 & 5 & 6 & 7 \\ 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

Elementele de pe diagonala secundară au proprietatea că suma indecșilor este $n-1$, adică $i+j=n-1$. Produsul poate fi calculat astfel:
$P = matrix[0][3] * matrix[1][2] * matrix[2][1] * matrix[3][0] = 7 * 5 * 3 * 1 = 105$
Răspunsul corect este **d)**.

59. Programul contorizează numărul de cifre ale tuturor elementelor vectorului, mai puțin elementele de valoare 0.
Răspunsul corect este **b)**.

60. Se parcurge matricea pe linii și pe coloane. Dacă *i* este impar se incrementează elementul. Dacă *j* este par se împarte la 2 elementul. Dacă ambele sunt adevărate se aplică incrementarea, urmată de împărțirea la 2. Fiecare element este actualizat în funcție de poziția sa în matrice.
Răspunsul corect este **a)**.

61. Matricea *matrix* este inițializată cu *0*. Prima buclă *for* va popula diagonala principală cu valori de *-1* și pe cea secundară cu valori de *1*. Întrucât instrucțiunea pentru diagonala secundară este a doua, la intersecția dintre cele două diagonale se va stoca valoarea *1*.
Al doilea *for* calculează suma elementelor matrice. Acestea sunt 0, exceptând diagonalele. Având în vedere că cele două au număr egal de elemente, una are valori de 1 și cealaltă de -1, suma este 0. Însă, având în vedere intersecția celor două diagonale, suma este dată de elementul din centrul matricei, adică 1.
Răspunsul corect este **b)**.

62. Vectorul *v* este inițializat cu *0* pentru fiecare element. Condiția din *if*, și anume *v[i]* va fi evaluată mereu ca fiind *false*, astfel, instrucțiunea `v[i]+=1` nu se va executa niciodată. Astfel, valorile din *v* vor rămâne {0,0,0,0,0}.

Răspunsul corect este **b)**.

63. Bucla *for* realizează parcurgerea vectorului din două în două elemente, începând de la elementul de pe poziția 1; în funcție de paritatea elementului curent, se calculează un status(care poate fi *0* sau *1*). Se adaugă *status* la elementul curent, iar dacă $status = 1$ atunci programul omite încă un element (i crește *3* în loc de *2*).

Răspunsul corect este **d)**.

64. Parcurgând instrucțiunile date cu valorile $a = 32$ și $b = 96$ putem observa că obținem chiar matricea cerută.

Răspunsul corect este **a)**.

65. Programul pornește de la o verificare a șirului *s* că este palindrom. Practic, variabila count va număra câte caractere nu respectă cerința de palindrom. De asemenea, *func* face și copierea caracterelor din jumătatea stângă în jumătatea dreaptă în ordine inversă (ultimul ia valoarea primului, penultimul celui de-al doilea, etc.). Pentru șirul *abcxa func* va afișa chiar *1 abcba.*

Răspunsul corect este **b)**.

66. Bucla *for* parcurge toate elementele vectorului. Condiția `i % 2` se evaluează la *true* atunci când *i* este impar, deci sunt considerate doar elementele de pe poziții impare. Condiția `v[i] % 2 == 1` verifică dacă elementul din *v* este impar.

Răspunsul corect este **c)**.

67. Se verifică pentru fiecare variantă în parte care va fi valoarea variabilei *x*. În urma execuției valoarea lui *x* va fi:

$$x = 10 + \frac{v[0]}{2} + v[1] + \frac{v[2]}{2} + v[3] = 10 + 6 + 17 + 4 + 13 = 50$$

Răspunsul corect este **a)**.

68. Ultimul element de pe coloana secundară a matricei reprezintă elementul $B[N-1][0]$, iar dacă înlocuim cu formula de calcul a elementului $\mathrm{B[N-1][0]} = (\mathrm{N}-1+1)^3 + (0+1)^3 = N^3 + 1$

Răspunsul corect este **a)**.

69. Se poate observa faptul că unele dintre elemente sunt suprascrise cu 4, după ce iau inițial valoarea *i * n + j * n*. Pentru a simplifica, putem alege unul dintre elemente, precum *a[1][2]*. Acesta va avea inițial atribuită valoarea 12, care ulterior va fi suprascrisă cu valoarea 4. Prin urmare, variantele b și c pot fi eliminate. Se verifică apoi element cu element valorile pentru varianta a.

Răspunsul corect este **a)**.

70. Se observă ușor că funcția verifică dacă elementele vectorului sunt în oglindă față de mijlocul vectorului (primul este egal cu ultimul, al doilea cu penultimul, șamd).

Răspunsul corect este **d)**.

71. Subprogramul completează toate pozițiile din matrice de pe diagonala principală și de deasupra acesteia. Numărul maxim de caractere se calculează cu suma lui Gauss: $50 + 49 + 48 + 47 + \ldots + 1 = 1275$.

Răspunsul corect este **c)**.

72. Programul calculează elementele de pe diagonala secundară. Dacă această sumă este număr prim, returnează 1. Pentru fiecare dintre variante, calculăm suma elementelor selectate și verificăm dacă suma este număr prim.

Răspunsul corect este **d)**.

73. Pentru fiecare element al vectorului vor fi realizate următoarele:
- se verifică dacă numărul este negativ. Dacă da, numărul nu este afișat.
- se verifică dacă numărul este pătrat perfect prin condiția `!(v[i] - j * j)`. Acestea sunt elementele afișate.

Răspunsul corect este **d)**.

74. Programul afișează subșirul cu suma maximă, calculează suma elementelor din acest subșir și afișează suma găsită. Într-un vector de numere întregi, suma maximă va fi formată din suma elementelor pozitive.
Pentru $v = \{4, -1, 2, 1, -5, 4\}$, elementele pozitive sunt $\{4, 2, 1, 4\}$ iar suma lor este *11*.

Răspunsul corect este **d)**.

75. Programul calculează suma elementelor de pe diagonala principală, din doi în doi și a elementelor de pe diagonala secundară, tot din doi în doi.

Răspunsul corect este **a)**.

76. Secvența de program modifică șirul inițial punând pe poziția i caracterul de pe poziția i +1. De remarcat faptul că apelul funcției *f* nu are impact asupra rezultatului programului întrucât incrementarea lui x este realizată după returnarea valorii.

Răspunsul corect este **b)**.

77. Cu ajutorul celor două instrucțiuni ciclice sunt parcurse doar elementele de sub diagonala principală a matricei (reprezintă elementele aflate deasupra diagonalei principale a matricei transpuse. Prin condiția impusă în instrucțiunea *if* sunt filtrate pentru afișare numai elementele divizibile și cu 2 și cu 5 adică divizibile cu 10.

Răspunsul corect este **c)**.

78. Prima instrucțiune ciclică inițializează prima linie și prima coloana la valoarea 2. Cele două instrucțiuni *for* imbricate realizează generarea fiecărui element din matrice, într-o formă iterativă, prin însumarea elementului de deasupra și a celui din stânga sa. Se generează o matrice simetrică fața de diagonala principală (afirmația *c*). Celelalte afirmații sunt false.

Răspunsul corect este **c)**.

79. Secvența de cod din cadrul funcției *my_func* parcurge toate elementele matricei iar elementele aflate pe diagonala secundară sunt interschimbate cu elementele corespunzătoare aflate pe diagonala principală de pe aceeași linie.

Răspunsul corect este **b)**.

80. Funcția *strstr* va căuta prima apariție a șirului *Militara* în șirul *Academia Tehnica Militara* și va returna adresa la care acesta apare. Funcția *printf* va afișa șirul de la respectiva adresă.

Răspunsul corect este **a)**.

81. Analizăm efectul fiecărei instrucțiuni, astfel:

`strcpy(aux, s1 + 2);` rezultă *aux = misATM*
`strcat(s1, aux + 3);` rezultă *s1 = AdmisATMATM*
`strcpy(s1, s1 + 5);` rezultă *s1 = ATMATM*

Răspunsul corect este **c)**.

82. Programul parcurge șirul și copiază doar caracterele care nu sunt spații, suprascriind șirul original. Astfel, se obține șirul inițial fără spații.

Răspunsul corect este **b)**.

83. Prin folosirea funcției *strchr* se verifică dacă există cel puțin o apariție a caracterului *ch* în *sir*. Modul în care este apelată funcția *strcpy* face să fie ștearsă prima apariție a caracterului *ch* din șir.). Făcând acest lucru în mod repetat, se ajunge ca în șirul *sir* să nu mai existe nici o apariție a caracterului *ch*.
Observație: `strcpy(s,s+1)` elimină primul caracter din șirul *s*

Răspunsul corect este **c)**.

84. Programul determină și afișează numărul de caractere din șirul *s*. Condiția `s[i]==0` verifică dacă parcurgerea a ajuns la terminatorul de șir.

Răspunsul corect este **a)**.

85. Programul parcurge șirul și face o serie de modificări fiecărui caracter, astfel: dacă este '.' sau ':', caracterul este înlocuit cu spațiu. Excepție face ultimul caracter din șir.

Răspunsul corect este **a)**.

86. Programul calculează suma caracterelor de tip cifră multiplu de *3* din cadrul șirului de intrare. Șirul de intrare conține caractere de la *'0'* la *'9'*, având coduri ASCII cuprinse între *0x30* și *0x39*. Suma cifrelor multiplu de *3* din intervalul $[0, 9]$ este *18*.

Răspunsul corect este **c)**.

87. Funcția schimbă conținutul șirului de intrare după cum urmează: caracterele: de tip cifră impară sunt înlocuite cu literele mici corespunzătoare. Exemplu: caracterul *'1'* devine *'a'*, caracterul *'3'* devine *'b'*, s.a.m.d.

Răspunsul corect este **c)**.

88. Instrucțiunea `strcpy(strchr(v,'a'), strchr(v,'a')+1);` elimină prima apariție a caracterului *'a'* din șirul *v*. Programul realizează această eliminare, până când nu mai există apariții ale caracterului *'a'* în *v*. În final, se afișează variabila *i*, adică de câte ori a fost eliminat un caracter *'a'* din *v*.

Răspunsul corect este **c)**.

89. Programul transformă literele majuscule, în litere minuscule, exceptând primul caracter. Conform tabelei ASCII, diferența dintre caracterele '*a'* și *'A'* este aceeași ca diferenața dintre *'b'* și *'B'*, etc. Tot conform tabelei ASCII, literele mari au valoarea asociată mai mică decât literele mici. Astfel atunci când adunăm unei litere mari diferența dintre 'a' și 'A', transformăm litera mare în literă mică.

Răspunsul corect este **b)**.

90. La fiecare apel al funcției *f,* se șterge al *i*-lea caracter. Astfel, la primul apel se șterge primul caracter, la al doilea apel se șterge al doilea caracter din șirul *rămas*. Pentru șirul "*admitere_atmFI*", pe ecran se vor afișa următoarele șiruri: *"admitere_atmFI", "dmitere_atmFI", "ditere_atmFI", "diere_atmFI", "diee_atmFI"*, etc. La a cincea afișare, se afișează șirul *"diee_atmFI"*.

Răspunsul corect este **d)**.

91. Bucla *for* realizează parcurgerea vectorului din două în două elemente, începând de la elementul de pe poziția *1*. Programul verifică dacă $sir[i]$ este literă mică. Dacă această condiție este respectată, atunci caracterul este copiat pe poziția anterioară în vector.

Răspunsul corect este **a)**.

92. Programul parcurge șirul de caractere dat și returnează suma totală, calculată prin adăugarea valorii *1* pentru fiecare literă mică, *2* pentru fiecare literă mare și *3* pentru fiecare caracter non-alfabetic.
În șirul de caractere *„Admitere2024ATM"* sunt: *7* litere mici, *4* litere mari și *4* caractere non-alfabetice. Astfel, $count = 7 * 1 + 4 * 2 + 4 * 3 = 27$.

Răspunsul corect este **b)**.

93. Funcția *strchr* caută un caracter într-un șir de caractere. Valoarea de return *a* lui strchr poate fi considerată *true* dacă găsește caracterul în șirul primit ca prim parametru sau *false* dacă nu găsește caracterul în șir. Astfel căutam caracterul curent ($sir[i]$) și cel anterior ($sir[i-1]$) în șirul de vocale *voc*. Dacă ambele căutări returnează *true*, înseamnă că avem *2* vocale consecutive și incrementăm variabila *count*.

Răspunsul corect este **a)**.

94. Orice caracter cu excepția *R, r, Y, y, g, G* nu au niciun efect. R copiază în șirul *str* mesajul *"Salut "*, suprascriind orice era înainte, apoi *'g'* concatenează la final mesajul George, obținându-se astfel mesajul dorit. Șirul *control* trebuie să conțină un *'R'* și un 'g', în această ordine pentru a putea genera șirul dorit.

Răspunsul corect este **a)**.

95. Șirului primit ca parametru i se modifică anumite caractere, fiind înlocuite cu *'*'*, în funcție de condiția `x % (y + i) == 0`. Putem analiza toate iterațiile astfel:

```
i=0, x % (y + i) = 24 % 3 = 0
i=1, x % (y + i) = 24 % 4 = 0
i=2, x % (y + i) = 24 % 5 = 4
i=3, x % (y + i) = 24 % 6 = 0
i=4, x % (y + i) = 24 % 7 = 3
i=5, x % (y + i) = 24 % 8 = 0
i=6, x % (y + i) = 24 % 9 = 6
i=7, x % (y + i) = 24 % 10 = 4
```

Unde rezultatul expresiei este *0*, caracterul se schimbă în *'*'*.

Răspunsul corect este **c)**.

96. Analizăm efectul fiecărei instrucțiuni care modifică cele două șiruri:
`strcpy(test1 + 2, test2)` – *test1* devine *„HeAtmFI"*
`strcpy(test2 + 3, test1)` – *test2* devine *„AtmHeAtmFI"*

Răspunsul corect este **b)**.

97. Programul parcurge toate elementele șirului și contorizează toate caracterele diferite de vocale (consoane și semne de punctuație, inclusiv spații).

Răspunsul corect este **c)**.

98. Programul parcurge toate elementele șirului și contorizează caracterele care nu sunt semne de punctuație. Chiar dacă se adună câte 2 elemente în variabila *counter*, variabila afișată este oricum împărțită la 2. Acestea sunt doar literele cuvintelor, adică 16.

Răspunsul corect este **d)**.

99. Programul parcurge toate elementele șirului și le afișează, cu câteva excepții. Toate caracterele semn de punctuație sunt omise, iar în loc de *'S'* este afișat *'N'*.
Obsevație: Atunci când imediat după *'.'* există un *'S'*, acesta nu va fi înlocuit cu *'N'* întrucât variabila *i* se va incrementa de *2* ori (o dată corespunzător `if(strchr(",.!?",s[i]))` și o dată corespunzător instrucțiunii `for`).

Răspunsul corect este **c)**.

100. Șirul *aux = "ATMFI"*, iar după instrucțiunea *s[8] = NULL*, șirul s devine *"admitere"*. Ulterior se șterg literele de pe pozițiile 3 și 4 din șirul *aux*, rezultând *aux = "ATM"*. În final se face concatenarea în *aux* rezultând "ATM-admitere".

Răspunsul corect este **d)**.

101. Secvența de program afișează elementele din vector aflate pe pozițiile indexate de valori întregi corespunzătoare caracterelor cifră din șirul de caractere dat.

Răspunsul corect este **c)**.

102. Programul inversează șirul, adică primul caracter este inversat cu ultimul, al doilea cu penultimul șamd.

Răspunsul corect este **b)**.

103. Pentru determinarea elementului maxim dintr-un vector sortat, nu este nevoie de parcurgerea vectorului. Fiind sortat, știm că ultimul element este maximul, adică obținem complexitate constantă: *O(1)*.

Răspunsul corect este **d)**.

104. Cele două bucle *for* sunt independente. Complexitatea primei bucle este logaritmică, $O(\log(n))$, iar complexitatea celei de-a doua este liniară, $O(n)$. Complexitățile fiind independente, considerăm complexitatea fiind cea superioară: $O(\log(n)) + O(n) = O(n)$

Răspunsul corect este **a)**.

105. Funcția *rec* se apelează recursiv pentru fiecare element al vectorului, deci numărul de apeluri va fi egal cu lungimea vectorului*(N)*.

Răspunsul corect este **c)**.

106. Algoritmul bubble sort are complexitate $O(n^2)$. Acesta este apelat de $\frac{n}{2}$ ori, *i* crescând din *2* în *2*. Astfel complexitatea secvenței de cod devine $O\left(n^2 * \frac{n}{2}\right) = O(n^3)$.

Răspunsul corect este **c)**.

107. Pentru a putea folosi căutarea binară într-un vector, acesta trebuie să fie ordonat.

Răspunsul corect este **b)**.

108. Bucla *for* realizează parcurgerea întregului șirul incrementând un contor de fiecare dată când găsește o paranteză deschisă. Dacă întâlnește o paranteză închisă verifică faptul că anterior a existat una deschisă. În cazul în care există, incrementează un al doilea contor pentru parantezele închise corect și decrementează numărul de paranteze deschise. de fiecare dată când se întâlnește o paranteză închisă care are una deschisă asociată. Pentru a afla răspunsul, trebuie numărate câte paranteze s-ar putea rezolva cu succes, dacă acestea ar fi parte dintr-o expresie matematică.

Răspunsul corect este **d)**.

109. Subprogramul func inversează elementele vectorului astfel: primul cu ultimul, al doilea cu penultimul, etc. Astfel, dintr-un vector ordonat crescător se obține un vector ordonat descrescător.

Răspunsul corect este **b)**.

110. Algoritmul selection sort caută mereu elementul de valoarea minimă din vector și îl așază pe prima poziție. Apoi repetă procedeul pentru restul vectorului. În acest caz concret, putem analiza cele două iterații, astfel:
Prima parcurgere:

- Se caută cel mai mic element din secvența [6,9,3,7,8,2].
- Cel mai mic element este 2, care se va schimba cu primul element 6.
- Secvența devine: [2, 9, 3, 7, 8, 6].

A doua parcurgere:

- Se caută cel mai mic element din secvența [9,3,7,8,6] (elementele de la poziția 1 încolo).
- Cel mai mic element este 3, care se va schimba cu 9.
- Secvența devine: [2, 3, 9, 7, 8, 6].

Răspunsul corect este **c)**.

111. Funcția *func* implementează recursiv o căutare binară într-un vector sortat, până la găsirea valorii căutate.

Răspunsul corect este **b)**.

112. Subprogramul ordonează crescător elementele din vector, conform unui algoritm bazat pe selection sort (este ales mereu elementul minim).

Răspunsul corect este **c)**.

113. Subprogramul *fun* implementează sortarea în ordine crescătoare a elementelor din *v,* folosind algoritmul Insertion Sort. Zona marcată cu ..... corespunde verificării condiției de intrare în secvența de cod în care elementele lui *v*, până la *i-1*, sunt deplasate dacă sunt mai mari decât *v[i]*. Prin urmare, elementele lui *v* se vor deplasa spre dreapta cât timp indexul *j* este mai mare decât zero, iar elementul curent *v[j]* este mai mare decât *v[i]*.

Răspunsul corect este **a)**.

114. Funcția *ex* determină intersecția intervalelor, selectând pentru partea stângă valoarea maximă posibilă, iar pentru partea dreaptă valoarea minimă posibilă.

Răspunsul corect este **d)**.

115. La fiecare iterație funcția afișează mereu cel mai mare număr din vector, după care asignează *-1* în locul acestuia în vector. Procesul este repetat de *n* ori, astfel afișându-se toate elementele vectorului în ordine descrescătoare.

Răspunsul corect este **c)**.

116. Funcția implementează algoritmul de căutare binară, prin urmare aceasta va returna indexul la care se regăsește elementul căutat.

Răspunsul corect este **c)**.

117. Programul realizează interschimbarea elementelor simetrice față de mijlocul vectorului, însă nu se oprește la mijlocul vectorului, astfel că vectorul revine la forma inițială.

Răspunsul corect este **d)**.

118. Întrucât vectorul este ordonat crescător iar *j* este mereu mai mare decât *i*, varianta de la punctul c este singura prin care poate se obține rezultatul dorit. Cu alte cuvinte, dacă diferența este prea mică, facem să crească j, iar în acest fel va crește și diferența.

Răspunsul corect este **c)**.

119. Având în vedere că toate elementele de pe o linie, exceptând exact unul singur sunt pare, înseamnă că suma elementelor de pe oricare linie este număr impar. Drept urmare, paritatea sumei tuturor elementelor este dată de numărul de linii, adică ar putea fi obținută folosind o instrucțiune de tipul *if(n%2)*.

Răspunsul corect este **a)**.

120. Subprogramul realizează interclasarea vectorilor *v* și *w*. Complexitatea temporală este $O(m+n)$.

Răspunsul corect este **a)**.

121. Buclele imbricate generează toate combinațiile de perechi distincte. Prima buclă parcurge *n−1* elemente, iar a doua buclă parcurge *n-i-1* elemente, rezultând o complexitate $O(n^2)$.

Răspunsul corect este **c)**.

122. Analiză complexitatea fiecărei bucle for, astfel:

for i: complexitate $O(n)$
for j: complexitate $O(n)$
for k: complexitate $O(\log_2(n))$
Cele trei bucle for sunt dependente de valoarea *n* în stabilirea iterațiilor realizate. Astfel complexitatea întregii secvențe reprezintă produsul complexității celor 3 bucle.

Răspunsul corect este **a)**.

123. Bucla exterioară *for* presupune *n* iterații. Pentru fiecare iterație a buclei exterioare se realizează *log(m)* iterații în bucla *for* interioară. Rezultă o complexitate de $O(nlog(m))$.

Răspunsul corect este **d)**.

124. Interclasarea vectorilor are complexitate $O(n)$.
OuickSort are complexitate $O(n\log(n))$.
Determinarea submulțimilor are complexitate exponențială $O(2^{\mathrm{n}})$
Adunarea matricelor presupune parcurgerea lor, deci $O(\mathrm{n} * \mathrm{n}) = O(n^2)$.
Cea mai redusă complexitate este cea a algoritmului de interclasare: $O(\mathrm{n})$

Răspunsul corect este **b)**.

125. Pentru determinarea elementului maxim dintr-un vector nesortat este nevoie de o singură parcurgere, adică *O(n)*.

Răspunsul corect este **a)**.

126. Ciclul for imbricat are complexitatea $O(\log_2(n))$, iar acesta se execută o sigură dată.
Observație: este modificată și valoarea variabilei *i*.

Răspunsul corect este **a)**.

127. Funcția recursivă parcurge prin variabila *n* toate valorile din intervalul [*0,n*]. Pentru fiecare, adună *n%2*, expresie care are valoarea *1* pentru numerele impare și *0* pentru numerele pare.

Răspunsul corect este **c)**.

128. Programul calculează în mod recursiv suma elementelor aflate pe diagonala principală și pe diagonala secundară.

Răspunsul corect este **a)**.

129. Subprogramul *p* implementează recursiv ridicarea la putere, unde *b* este baza și *e* este exponentul.

Răspunsul corect este **c)**.

130. Se va apela recursiv funcția *f*, cu *luna + 1* pentru a ști în ce lună ajungem cu recursivitatea, *buget_ramas - cheltuiala* reprezintă bugetul rămas pentru luna următoare, iar *2*cheltuiala* reprezintă cheltuielile pentru luna următoare. Continuăm recursivitatea până când nu mai sunt suficienți bani rămași pentru luna respectivă.

Răspunsul corect este **c)**.

131. Funcția parcurge fiecare caracter din șirul *s*. Funcție se comportă diferit în funcție de tipul caracterului: pentru litere se face apel recursiv și apoi afișarea caracterului, iar pentru cifre invers. Efectul este că toate cifrele se vor afișate la început în ordinea în care apar în *s*, iar literele sunt afișa la final, în ordinea inversă a apariției în lor *s*.

Răspunsul corect este **c)**.

132. Funcția se apelează recursiv până când *x* devine *0*. Afișările încep la finalul recursivității, deci când *x* are valoarea 0 și nivel are valoarea cea mai mare. Prin urmare, pe prima linie este afișată o singură steluță și mai multe spații. Pe a doua linie sunt *3* steluțe și cu un spațiu mai puțin, etc. Pentru a afișa 4 linii, așa cum este menționat în cerință, x trebuie să fie 4.

Răspunsul corect este **c)**.

133. Funcția se va apela recursiv, i având valori între 1 și n. Pentru fiecare apel, s va fi incrementat de i ori, adică va avea valoarea i la finalul buclei for. Astfel, programul calculează de fapt suma *1+2+3+...+n*, pentru *n=1024*. Rezultă s = 524800.

Răspunsul corect este **a)**.

134. Funcția calculează numărul de apeluri recursive până când a == b (diferența dintre a și b este 500 însă a crește cu 2, iar b scade cu 2, prin urmare vom avea 125 de apeluri ale funcției. Rezultatul este 125 + (750 + 750) /2 = 875.
Altfel explicat, după k apeluri ale lui f, valoarea apelului f(500, 1000) este k + f(500+2k, 1000-2k). Condiția de ieșire din recursivitate (a = b) are loc la pasul k dacă 500+2k = 1000-2k, de unde rezultă k = 125. Așadar, rezultatul final este 125 + f(500+250, 1000-250) = 875.

Răspunsul corect este **c)**.

135. Funcția *f* calculează recursiv al *n*-lea termen din șirul lui Fibonacci. Funcția *g* caută primul număr din șirul lui Fibonacci care să fie divizibil cu *2* și cu *3*.

136. Se analizează apelurile recursive ale funcției pentru a determina rezultatul. Este de remarcat faptul că după primele 3 afișări, singura soluție care se potrivește este chiar cea corectă.

Răspunsul corect este **b)**.

137. Funcția recursivă parcurge prin variabila *n* toate valorile din intervalul [*0,n*]. Numerele adunate sunt doar cele care verifică `n%10==(n/10)%10`, adică ultimele două cifre să fie egale.

Răspunsul corect este **a)**.

138. Funcția verificare verifică dacă șirul este același de la stânga la dreapta și invers. Șirul *"radar"* este un palindrom, deci programul va returna valoarea *1*.

Răspunsul corect este **d)**.

139. Funcția *f* identifică recursiv cel mai mare element dintr-un vector. Algoritmul compară ultimul element cu cel mai mare element determinat recursiv din restul vectorului.

Răspunsul corect este **d)**.

140. Subprogramul recursiv *f* returnează oglinditul numărului trimis ca parametru.

Răspunsul corect este **a)**.

141. Cât timp n este mai mare decât -10, funcția recursivă va afișa ultima cifră. Apoi, *n* este incrementat de două ori. Dacă ultima cifră este pară, se elimină ultimele 2 cifre, altfel se elimină o singură cifră. Recursivitatea se apelează cu *n-7*.

func(874576329) -> func(87457626) -> func(874569) -> func(87450) -> func(867) -> func(79) -> func(1) -> func(-7)

Răspunsul corect este **a)**.

142. Funcția *rec* prelucrează recursiv cifrele lui *x*. Pentru cifre *pare* adună *1*, iar pentru cifre *impare* adună *1000*.

Rezultatul final va fi $1000 * nr_{cifre\ impare} + 1 * nr_{cifre\ pare} = 5004$.

Răspunsul corect este **d)**.

143. Programul calculează suma cifrelor unui număr întreg și aplică recursiv această sumă până când rezultatul este un număr mai mic decât *10*. Se calculează suma cifrelor până se ajunge la o singură cifră. Apelurile sunt realizate astfel: $f(9875) -> f(29) -> f(11) -> f(2)$.

Răspunsul corect este **d)**.

144. Programul calculează inversul numărului obținut din cifrele divizibile cu 3 ale parametrului *x*. Verificarea din *main* face ca mereu să se afișeze YES.

Răspunsul corect este **b)**.

145. Apelul *f(3,5)* va returna *f(4,3)+f(2,4) = 4+5 = 9*

f(4,3) va returna *f(2,3) + f(2,2) = 3+2=5;*

f(2,4) va returna *4*.

Răspunsul corect este **c)**.

146. Se apelează funcția *F* cu valorile *n = 4, m = 2* și *k = 3*.

În primul apel, *n* este diferit de *0*, deci se calculează *1* ori *2* și se apelează *F* cu *n = 3, m = 3* și *k = 3*. Se obține *2 + F(3, 3, 3)*.

În al doilea apel, n este diferit de 0, deci se calculează *0* ori *3* și se apelează *F* cu *n = 2, m = 4* și *k = 3*. Se obține *0 + F(2, 4, 3)*.

În al treilea apel, *n* este diferit de *0*, deci se calculează *2* ori *4* și se apelează *F* cu *n = 1, m = 5* și *k = 3*. Se obține *8 + F(1, 5, 3)*.

În al patrulea apel, *n* este diferit de *0*, deci se calculează *1* ori *5* și se apelează *F* cu *n = 0, m = 6* și *k = 3*. Se obține *5 + F(0, 6, 3)*.

În al cincilea apel, *n* este *0*, deci se returnează m, adică *6*.

Se urcă înapoi în apeluri. Al patrulea apel devine *5 + 6*, adică *11*.

Al treilea apel devine *8 + 11*, adică *19*.

Al doilea apel devine *0 + 19*, adică *19*.

Primul apel devine *2 + 19*, adică *21*.

Răspunsul corect este **a)**.

147. Programul afișează șirul de caractere în ordine inversă. Incrementarea indexului în șir se face înaintea apelului funcției *fun*, astfel, după ieșirea din *fun* ultima valoare afișată va fi $str[x + 1]$, unde x este valoarea celui de al doilea argument al apelului.

Răspunsul corect este **a)**.

148. Funcția *func* realizează operații în funcție de restul împărțirii lui *n* la *3*. Apelul *func(32)* desfășoară recursiv calculele $1 + 2 + 2 + 5 = 10$.

Răspunsul corect este **c)**.

149. Se analizează fiecare apel recursiv al funcției și se calculează valoarea returnată, astfel:
*iteratia 1: x=6 y=4, returnează **2**+6-4=**4***
*iteratia 2: x=3 y=4, returnează **3**+3-4=**2***
*iteratia 3: x=3 y=1, returnează **1**+3-1=**3***
*iteratia 4: x=0 y=1, returnează **1***
Observație: a se observa că funcția este apelată cu parametrii x și y inversați între ei.

Răspunsul corect este **c)**.

150. Subprogramul contorizează recursiv numărul de cifre pare. Putem remarca faptul că apelul recursiv este identic *f(x/10)*, însă 1 este adăugat numai atunci când numărul este par.

Răspunsul corect este **c)**.

151. Funcția întoarce cel mai mare număr de 2 cifre ce poate fi format din cifrele numărului *n* luate câte două(ordinea cifrelor se păstrează), astfel: 79, 17, 81, 18, 51, 25. Valoarea returnată în final este 81.

Răspunsul corect este **c)**.

152. Se observă că funcția întoarce suma cifrelor unui număr întreg. Numărul *1040212* este singurul din cele furnizate care au suma cifrelor egală cu 10.

Răspunsul corect este **b)**.

153. Subprogramul *fun* calculează recursiv lungimea șirului de caractere *v*. Este nevoie ca variabila i să crească la fiecare apel recursiv. Altfel, s-ar obține o buclă infinită. De asemenea, funcția trebuie să returneze 1+apelul recursiv. Altfel nu s-ar mai contoriza caracterele și valoarea returnată în final ar fi 1.

Răspunsul corect este **b)**.

154. Linia 4 este greșită deoarece atunci când *n* rămâne nemodificat, recursivitatea nu se poate încheia.

Răspunsul corect este **a)**.

155. Subprogramul returnează indexul primei apariții a caracterului *c* în șirul *str*, sau *-1* dacă *c* nu există în *str*. În subprogram nu există verificarea parcurgerii întregului șir. Prin urmare, în zona marcată cu . . . . . trebuie să verifice faptul că am terminat de parcurs șirul de caractere.

Răspunsul corect este **a)**.

156. Singura afirmație adevărată din cele 4 este prima: poate fi utilizat pentru a genera toate soluțiile unei probleme.

Răspunsul corect este **b)**.

157. Numărul *337* este un prim permutabil deoarece toate permutările sale *(337, 373 și 733)* sunt numere prime. Și *373* este prim permutabil, însă este mai mare decât *337*, prin urmare va fi generat ulterior.
317 și *371* nu sunt prime permutabile pentru că *371=7*53*.

Răspunsul corect este **c)**.

158. Maria generează primele *12* numerele diferite de *5* cifre, în ordine crescătoare folosind combinațiile cifrelor date după cum urmează: *13366, 13636, 13663, 16336, 16363, 16633, 31366, 31636, 31663, 33166, 33616,* ***33661***

Răspunsul corect este **c)**.

159. Numerele generate sunt: 135 144 153 225 234 243 **252** 315 324 333 342 351 405 414 423 432 441 450 504 513 522 531 **540.**
Nu este neapărat necesară generarea tuturor numerelor pentru a găsi răspunsul corect. Pornind de la cele 3 numere din enunț, se generează doar următoarele 4. Referitor la ultimul element, putem considera că 540 este cel corect întrucât după acesta, conform regulilor de generare nu mai putem genera alt număr.

Răspunsul corect este **c)**.

160. Respectând constrângerile, se pot genera următoarele secvențe:

a c e b	a d f b	a f b d
a c f b	a d f c	a f b e
a c f d	a e b d	a f c e
a d b e	a e b f	a f d b.
a d b f	a e c f	

Răspunsul corect este **c)**.

161. Programul folosește un algoritm de tip backtracking pentru a genera toate permutările unui șir de caractere. Funcția *f2* parcurge fiecare caracter al șirului și îl plasează pe diferite poziții, cu ajutorul funcției *f1*, în mod recursiv, pentru a explora toate posibilitățile. După fiecare permutare găsită, se revine la starea anterioară a șirului și se încearcă alte permutări (funcția *f* este apelată de 2 ori)

Răspunsul corect este **c)**.

162. Matricea de adiacență a unui graf neorientat cu *n* noduri și *m* muchii conține câte două valori de 1 pentru fiecare muchie: una deasupra diagonalei principale și una dedesubt. Restul elementelor sunt *0*. Suma elementelor din matrice este egală cu numărul de muchii*2, deci răspunsul este *m*2=50*.

Răspunsul corect este **b)**.

163. Matricea de adiacență a unui graf cu *n* noduri este o matrice pătratică de dimensiune *n×n*, unde fiecare linie și coloană corespunde unui nod. Pentru *n=20*, dimensiunea matricei este *20 x 20*

Răspunsul corect este **a)**.

164. Graful V are nodurile (1, 2, 3) care au câte 2 vecini.

Răspunsul corect este **c)**.

165. Graful are 5 noduri dintre care nodurile 0 și 4 au grad 1 iar nodurile 1, 2, 3 au grad 2. Gradul cumulat este 8

Răspunsul corect este **c)**.

166. Graful are 1 vârf izolat, respectiv vârful 4.

Răspunsul corect este **c)**.

167. Un graf complet cu 5 noduri conține $\frac{n(n-1)}{2} = 10$ muchii. Graful complementar lui G conține toate muchiile din graful complet, mai puțin muchiile lui G. Astfel, $nr_{complementar} = nr_{complet} - nr_g = 10 - 2 = 8$.

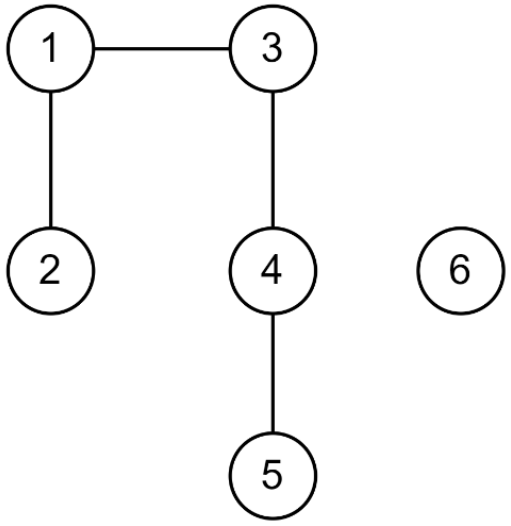
Răspunsul corect este **c)**

168. Analizăm fiecare afirmație, astfel:

- Graful este conex. *Fals*. Graful are 2 componente conexe 0-1-2 și 3-4
- Nodul 1 are gradul 2. *Adevărat*. Pe a doua linie din matrice găsim 2 valori de 1.
- Graful conține un ciclu. *Adevărat*. Ciclul din graful de față este 0-1-2.
- Matricea de adiacență este simetrică. *Adevărat*. Orice graf neorientat respectă această proprietate.

Răspunsul corect este **a)**.

169. Graful poate fi reprezentat astfel:



Vom verifica fiecare dintre afirmații, pe rând, astfel:

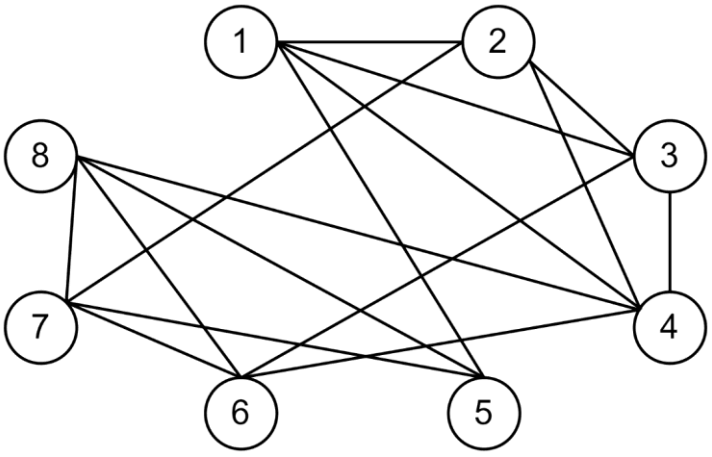
a. Nodul *6* este izolat - *adevărat*.
b. Adăugând muchiile *(5,6)* și *(6,2)* putem obține un ciclu hamiltonian în graf, prin urmare graful devine hamiltonian – *adevărat*.
c. Adăugând muchia *(5,2)* putem obține un graf eulerian – *fals*.
d. Din reprezentarea grafică se observă ușor că graful are exact *2* componente conexe – *adevărat*.

Răspunsul corect este **c)**.

170. Nodul 2 are conexiune prin muchii cu alte 3 noduri, deci gradul său este 3.

Răspunsul corect este **d)**.

171. Graful poate fi reprezentat grafic astfel:



Graful este conex deoarece există un drum între oricare două noduri.
Graful este hamiltonian deoarece există un ciclu care trece prin toate nodurile exact o dată (exemplu: *1→2→3→4→6→7→8→5→1*).
Graful nu este eulerian deoarece nu toate nodurile au grade pare (nodul *5* are grad impar).

Răspunsul corect este **a)**.

172. Având în vedere că un nod oarecare este mereu izolat, atunci putem reprezenta pe rând cele *5* noduri ca fiind izolate, generând ulterior un graf de *4* noduri. Numărul maxim de grafuri posibile se rezumă la *5* * numărul de grafuri cu 4 noduri. Numărul de grafuri care poate fi reprezentat este de $2^{numar\ muchii}$, iar numărul maxim al muchiilor pentru un graf cu 4 noduri este de $\frac{4(4-1)}{2} = 6$. Astfel, numărul maxim de grafuri ce ar putea fi reprezentat este $5 * 2^6$.
Totuși, putem observa cum pentru un graf din cele 2^6 toate cele 4 noduri sunt izolate, prin urmare, această configurație de graf trebuie luată în calcul o singură dată, și nu de 5 ori. Astfel, numărul de grafuri care poate fi generat respectând condițiile date este mai mic decât $5 * 2^6$.

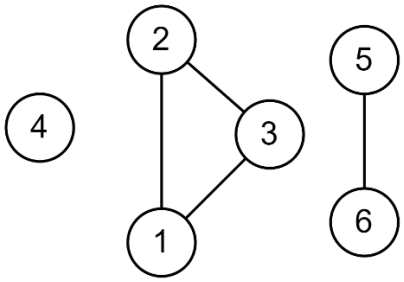
Răspunsul corect este **d)**.

173. Graful este neorientat => suma gradelor tuturor nodurilor trebuie să fie număr par. Putem elimina varianta *d*.
Graful este conex => nu putem avea elemente izolate. Cum în varianta a, nodul *5* este izolat, înseamnă că nu poate reprezenta un graf conex. Putem elimina varianta *a*.
Pentru a conecta toate nodurile dintr-un graf avem nevoie de minimum n-1 muchii. Pentru varianta *b*, numărul de muchii este *4 (suma gradelor/2)*. Nu putem construi un graf conex, întrucât sunt prea puține muchii.
Varianta *b* respectă toate condițiile enunțate anterior.

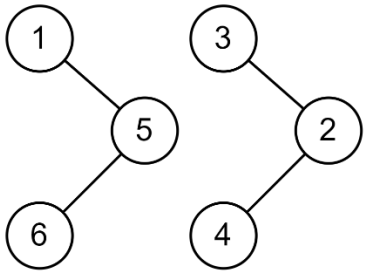
Răspunsul corect este **b)**.

174. Se reprezintă grafic cele 4 grafuri.

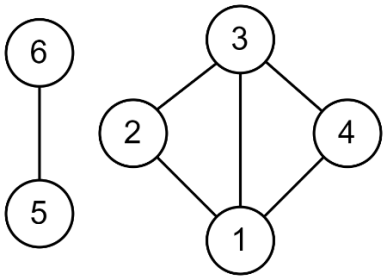
a.



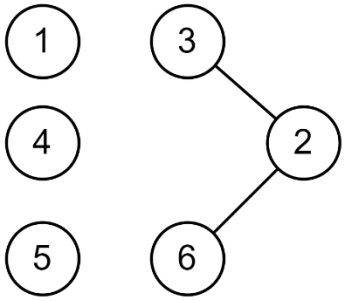
c.



b.



d.



Răspunsul corect este **a)**.

175. Poate fi reprezentat grafic graful și apoi se numără nodurile terminale. De asemenea, un nod terminal are proprietatea că gradul lui este *1*. Astfel putem număra câte noduri au gradul 1. Numerotând nodurile de *1* la *7*, nodurile cu gradul *1* sunt: *4, 6, 7*.

Răspunsul corect este **c)**.

176. Subgraful conține vârfurile *1, 3, 4, 7*. Un graf este complet dacă oricare *2* vârfuri distincte ale sale sunt adiacente. Inițial subgraful are muchiile: *[1, 4]* și *[1, 3]*. Pentru a deveni graf complet este nevoie să adăugăm următoarele muchii: *[1, 7], [3, 4], [3, 7], [4,7]*.

Răspunsul corect este **a)**.

177. Se reprezintă grafic graful și se numără muchiile. De asemenea, graful fiind neorientat, fiecărei muchii îi corespund în matricea de adiacență două valori de *1*. Astfel, numărul de muchii este egal cu suma elementelor matricei de adiacență împărțită la 2.

Răspunsul corect este **b)**.

178. Dacă gradul nodului 1 este *x* atunci suma elementelor de pe linia 1 trebuie să fie *x*. Analizăm fiecare variantă în parte. Numai pentru varianta *a,* suma pe linii este *2, 2, 1, 3.*

Răspunsul corect este **a)**.

179. După construcția grafică a fiecărui graf analizăm fiecare variantă:
- graful din varianta *a* are o singură componentă conexă
- grafurile *b* și *c* au câte două
- graful *d* are 3 componente conexe.

Răspunsul corect este **d)**.

180. Funcția *test* afișează suma elementelor matricei de adiacență. Având în vedere că o muchie are o contribuție de *2* în matricea de adiacență (graful neorientat având o matrice simetrică), singura afirmație certă despre graful *G* este că are *10/2=5* muchii.

Răspunsul corect este **a)**.

181. Un graf este conex dacă există un drum între oricare două noduri. Pornind de la orice nod, toate celelalte noduri sunt accesibile folosind BFS/DFS, deci graful este conex.

Răspunsul corect este **a)**.

182. Gradul unui nod este numărul de muchii care pleacă din acel nod. Pentru a determina gradul fiecărui nod, pot fi numărate valorile de 1 de pe linia corespunzătoare nodului.
Nodul 0 are două muchii. Grad = 2.
Nodul 1 are trei muchii. Grad = 3.
Nodul 2 are trei muchii. Grad = 3.
Nodul 3 are două muchii. Grad = 2.
Nodul 4 are două muchii. Grad = 2

Răspunsul corect este **a)**.

183. Matricea de adiacență este $N \times N$, deci conține N^2 elemente. Întrucât graful este neorientat (simetric față de diagonala principală) și pentru că pe diagonala principală valoarea este *0*, avem $\frac{N(N-1)}{2}$ elemente de interes. Având în vedere că fiecare element are 2 valori posibile, putem obține $2^{\frac{N(N-1)}{2}}$ posibilități. Faptul că selectăm ca una dintre valori să fie 1 sau 0 este echivalent cu a selecta ca muchia să fie prezentă sau nu în graf.

Răspunsul corect este **b)**.

184. Matricea de adiacență are dimensiunea de *10×10*. Întrucât graful este neorientat (simetric față de diagonala principală) și pentru că pe diagonala principală valoarea este *0*, avem $\frac{N(N-1)}{2}$ elemente de interes, adică 45, din care putem alege. Numărul de grafuri parțiale este: $\binom{45}{3}$, combinări de 45 luate câte 3.

Răspunsul corect este **d)**.

185. Matricea de adiacență are dimensiunea de *20×20*. Întrucât graful este neorientat (simetric față de diagonala principală) și pentru că pe diagonala principală valoarea este *0*, avem $\frac{N(N-1)}{2}$ elemente de interes, adică *190*, din care putem alege. Pentru a determina numărul de grafuri parțiale cu maximum 2 muchii trebuie să calculăm: $\binom{190}{0} + \binom{190}{1} + \binom{190}{2} = 18146$.

Răspunsul corect este **a)**.

186. Muchiile respectă relația $y = 2x + 3$, unde $x, y \in \{1, 2, \ldots 1000\}$. Pentru valoarea minimă a lui *x*, adică $x = 1 \Rightarrow y = 5$. Pentru $x = 2 \Rightarrow y = 7$. Pentru $x = 498 \Rightarrow y = 999$. Pentru $x = 499 \Rightarrow y = 1001$, care depășește *1000*. Cum funcția $2x + 3$ este una crescătoare monotonă, nu există nici o muchie pentru nodul 499 și nodurile mai mari decât acesta, iar toate muchiile care pornesc din nodurile $1, 2, \ldots, 498$ există.

Răspunsul corect este **a)**.

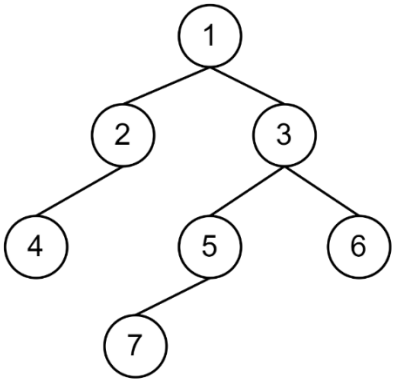
187. Fiecare nod al grafului are două noduri adiacente, deci toate nodurile au același grad (*2*).

Răspunsul corect este **d)**.

188. Arborele binar cu numărul minim de niveluri este un arbore în care toate nodurile de pe nivelurile intermediare au exact *2* fii, adică este un arbore cât mai compact. Astfel, fiecare nivel fiind complet, un arbore binar cu 5 niveluri conține $2^0 + 2^1 + 2^2 + 2^3 + 2^4 = 31$ noduri.

Răspunsul corect este **b)**.

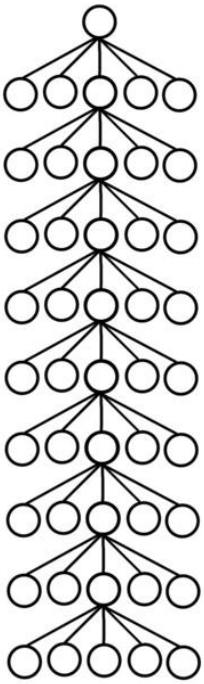
189. Un nod fără descendenți este un nod frunză. Conform reprezentării grafului, avem 3 noduri terminale: 4, 6 și 7.



Răspunsul corect este **c)**.

190. Analizăm fiecare parcurgere a arborelui, astfel:
Parcurgerea în inordine stânga – vârf – dreapta: 5 2 3 1 4 6
Parcurgerea în preordine vârf – stânga – dreapta: 1 2 5 3 4 6
Parcurgerea în postordine stânga – dreapta – vârf: 1 2 5 3 4 6
Răspunsul corect este **d)**.

191. Pentru a obține în arborele dat o înălțime maximă, trebuie să așezăm nodurile frunză cât mai în adâncime. Pentru a reuși o astfel de abordare, trebuie ca la fiecare nivel să avem 4 noduri frunză și unul părinte pentru alte 5 noduri. Cu aceasta modalitate, obținem înălțimea maximă ca fiind 9. Arborele poate fi reprezentat astfel:

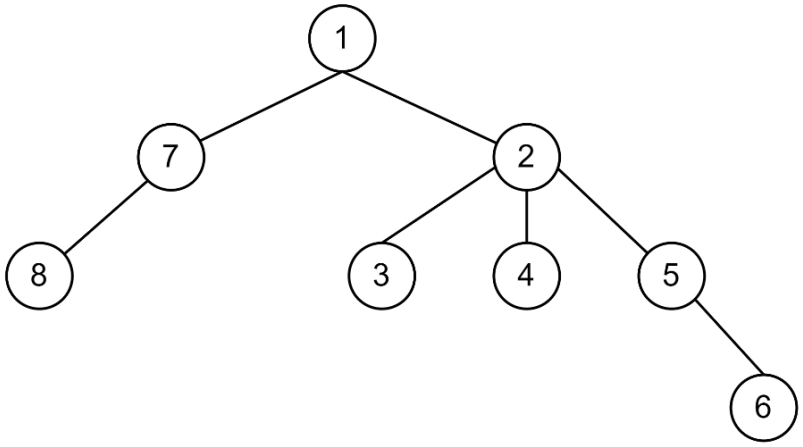


Răspunsul corect este **a)**.

192. Elementele care nu apar în vectorul de tați nu au descendenți în arbore, prin urmare sunt noduri de tip frunză. Aceste noduri sunt: *1, 4, 5, 8* și *9*.

Răspunsul corect este **b)**.

193. Arborele poate fi reprezentat grafic astfel:



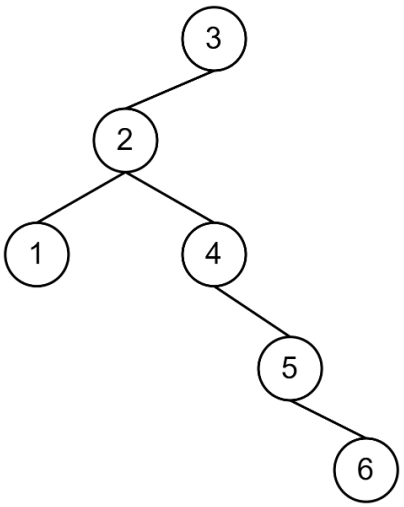
Se poate observa că distanța cea mai mare se regăsește între nodurile 8 și 6, prin urmare, ar trebui să reconstruim arborele astfel încât unul dintre cele două noduri să fie rădăcină.
Pentru nodul *8* rădăcină, vectorul de părinți devine $[7, 1, 2, 2, 2, 5, 8, 0]$
Pentru nodul *6* rădăcină, vectorul de părinți devine $[2, 5, 2, 2, 6, 0, 1, 7]$.
Dintre ele, doar varianta cu rădăcina 6 se regăsește în variantele de răspuns.

Răspunsul corect este **a)**.

194. Analizând lista de muchii putem observa următoarele:
- Nodul *4* are *2* descendenți direcți *(fii)*: nodurile *6* și *11*
- Înălțimea subarborelui cu rădăcina *3* este dată de muchiile *[3,4]*,*[4,11]*.

Răspunsul corect este **d)**.

195. Se reprezintă grafic arborele pe baza matricei de adiacență.



Nodul *1* nu are noduri de tip copil.

Răspunsul corect este **a)**.

196. Numărul maxim de frunze se obține în cadrul unui arbore binar cu fiecare nivel intermediar completat cu noduri. Numărul maxim de frunze poate fi calculat astfel: $\left[\frac{N+1}{2}\right] = \left[\frac{63+1}{2}\right] = 32$.

Răspunsul corect este **d)**.

197. Nodul 5 este tatăl cel mai des întâlnit în vectorul de tați astfel că din el vor pleca cele mai multe noduri (deci gradul cel mai mare). Gradul unui nod într-un arbore dat prin vectorul de tați este egal cu numărul de apariții, la care se adaugă 1 dacă nodul nu este rădăcină. Chiar dacă nodul 5 este rădăcină, el apare de 4 ori în vector, pe când al doilea cel mai frecvent nod apare de 2 ori, adică are gradul 3.

Răspunsul corect este **c)**.

PROBLEME DE NIVEL MEDIU-DIFICIL. ENUNȚURI INSTRUCȚIUNI ȘI OPERATORI

1. Ce afișează secvența de cod de mai jos?

```
int i = 40, j = 42;
int a[] = { 238,  28,  870,  9,  465,  3  };
printf("%d", a['j' - 'i']);
```

a) 238 **b)** 870 **c)** 28 **d)** Eroare de compilare

2. Se consideră subprogramul de mai jos. Ce va fi afișat în urma execuției apelului *f(100, 8)*?

```
void f(int numar, int baza){
      char alfabet[] = "0123456789ABCDEF";
      char reprezentare[100];
      int i = 0;
      while (numar){
            reprezentare[i++] = alfabet[numar % baza];
            numar /= baza;
      }
      i--;
      for (; i >= 0; i--)
            printf("%c", reprezentare[i]);
}
```

a) 202 **b)** 144 **c)** 64 **d)** Nu va fi afișat nimic

3. Se consideră a, un număr întreg cu proprietatea $10 \leq a \leq 1000$ și b un număr întreg pozitiv, nenul, par cu proprietatea $b \leq 100$. Care dintre următoarele condiții este adevărată pentru toate valorile ce respectă constrângerile menționate anterior?

a) *(a * b) > 0 && (a % b == 0)*
b) *(a - b > 0) || (b - a > 0)*
c) *(a * b) / 10 > 0) && (a / 10 - b < 100)*
d) *(a != b) || (b - a > 0)*

4. Se consideră definite trei variabile de tip *int: a, b* și *c*. Expresia care are valoarea *1* dacă și numai dacă exact două dintre variabile au valori identice este:

a) *a == b || b == c || a == c*
b) *(a == b && b != c) || (a == c && b != c) || (b == c && a != b)*
c) *a == b && a == c*
d) *!(a != b || a != c)*

5. Ce afișează programul de mai jos?

```
int operatie(int a, int b, int cod) {
	switch (cod) {
		case 1:
			return a + b;
		case 2:
			return a - b;
		default:
			return -888;
	}
}
int main() {
	int x = 10, y = 3, oper = 3, rezultat;
	if (oper >= 1 && oper <= 2)
		rezultat = operatie(x, y, oper);
	else
		rezultat = -999;
	printf("%d", rezultat);
}
```

a) 13 **b)** 7 **c)** -888 **d)** -999

6. Se consideră subprogramul de mai jos. Ce afișează acesta?

```
void fun(int n) {
	int f[10] = { 0 };
	while (n > 0) {
		f[n % 10]++;
		n /= 10;
	}
	int c = 0, max = 0, i;
	for (i = 0; i < 10; i++) {
		if (f[i] > max) {
			max = f[i];
			c = i;
		}
	}
	printf("%d", c);
}
```

a) cea mai mare cifră a lui n
b) numărul de cifre impare ale lui n
c) numărul de cifre pare ale lui n
d) cea mai frecventă cifră a lui n

7. Se consideră subprogramul de mai jos, unde *x* reprezintă un număr natural. Menționați ce vor afișa apelurile funcției *f*, pentru *x=83*, *x=2025*, *x=6891* și respectiv *x=523*.

```
void f(unsigned int x)
{
      int i, j;
      int v[10000]={0};
      v[0] = v[1] = 1;

      for (i=2; i*i<10000; i++)
            if (v[i] == 0)
                  for (j=2*i; j<10000; j+=i)
                        v[j] = 1;

      printf("%d ", v[x]);
}
```

a) 0 1 1 0 **b)** 1 0 0 1 **c)** 0 1 0 1 **d)** 1 0 1 0

8. Ce va afișa programul de mai jos în urma apelului *f(2924651)* ?

```
void f(int x)
{
      int rank = 1;
      int count = 0;

      while (x / (rank * 100) > 0)
      {
            if (x % (rank * 100) / rank > 50)
                  count++;

            rank *= 10;
      }
      printf("%d", count);
}
```

a) 92 **b)** 3 **c)** 51 **d)** 1

9. Ce condiție trebuie introdusă în bucla *while* astfel încât programul următor să calculeze valoarea cu două zecimale care aproximează cel mai bine rădăcina pătrată a lui *x*?

```
float modul(float x) {
	if (x < 0)
		return -x;
	return x;
}
float f(int x) {
	float aux = 1.01;
	float last = 1;
	while (.............) {
		last = aux;
		aux = aux + 0.01;
	}
	return last;
}
int main() {
	printf("%.2f", f(123));
}
```

a) `aux * aux != x`
b) `modul(x - aux * aux) > 0`
c) `modul(x - aux * aux) < modul(x - last * last)`
d) `modul(x - last * last) > 0`

10. Menționați ce afișează programul de mai jos.

```
void main() {
	long n = 345643, c=1, p = 1, s = 0, x, div;
	while (n / p) {
		x = 0;
		div = 1;
		for (int i = 1; i <= c; i++) {
			x += n / div % 10 * div;
			div *= 10;
		}
		if (c % 2)
			s += x;
		else
			s -= x;
		p *= 10;
		c++;
	}
	printf("%li", s);
}
```

a) 122320 **b)** -305040 **c)** -680280 **d)** 925820

11. Fie funcția *f* definită mai jos. Indicați ce returnează aceasta.

```
int f(int x, int y)
{
      int i, z = 0;
      for (i = 0; i < y; i = i += 2)
            z += x;
      for (i = 0; i < x * y; i++)
            z += y;
      return z;
}
```

a) $2x + \frac{y}{2} + y^2$ **b)** $\frac{x*y}{2} + x * y^2$ **c)** $\frac{x}{2} + y^2$ **d)** $\frac{x}{2} + x * y$

12. Ce valoare trebuie atribuită variabilei *b* pentru ca programul următor să afișeze valoarea *2*?

```
void main() {
      int a = 121;
      int b = . . . . . ;
      int saved_a = a;
      int saved_b = b;
      int count = 0;

      while (saved_b > 0) {
            if (a == 0) {
                  count++;
                  a = saved_a;
                  saved_b = saved_b / 10;
                  b = saved_b;
            }
            else {
                  if (a % 10 == b % 10) {
                        a = a / 10;
                        b = b / 10;
                  }
                  else {
                        a = saved_a;
                        saved_b = saved_b / 10;
                        b = saved_b;
                  }
            }
      }
      printf("%d", count);
}
```

a) 612121245 **b)** 121892212 **c)** 373121129 **d)** 111122222

13. Ce afișează programul de mai jos?

```
int pow(int m, int n){
      int aux = 1;
      for (int i = 0; i < n; i++)
            aux = aux * m;
      return aux;
}

void main(){
      int matrix[100][100];
      int i, j, m=29, n=30;

      for (int i = 0; i < m; i++)
            for (int j = 0; j < n; j++)
                  matrix[i][j] = pow(2, i + j);

      int sum = matrix[4][0]+matrix[1][3]+matrix[5][2];
      printf("%d", sum);
}
```

a) 292 **b)** 580 **c)** 160 **d)** 224

14. Fie funcția de mai jos. Indicați care vor fi valorile elementelor matricei *mat* după apelul *f(mat, 4)*.

```
void f(int mat[10][10], int n){
      int mat[10][10], i, j;

      for (i = 0; i < n; i++)
            for (j = 0; j < n; j++)
                  mat[i][j] = (i*i + 2*i*j + j*j) % n;
}
```

a) $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 3 & 2 & 3 & 2 \end{pmatrix}$

b) $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 3 & 0 & 3 \\ 0 & 2 & 2 & 0 \end{pmatrix}$

c) $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$

d) $\begin{pmatrix} 0 & 1 & 0 & 3 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 3 \\ 3 & 2 & 3 & 0 \end{pmatrix}$

15. Fie secvența de program de mai jos, unde *a* este o matrice cu *7* linii și *7* coloane, numerotarea liniilor și coloanelor începând de la *1*. Ce reprezintă valoarea din variabila *s* după executarea acestei secvențe?

```
int n = 7;
int i, j, s = 0;
for (i = 1; i <= n; i++)
      for (j = 1; j <= n; j++)
            if ((i+j)%2 == 0 && (i!=j) && (i+j != n+1))
                  s = s + a[i][j];
```

a) Suma tuturor elementelor aflate pe poziții cu suma indicilor pară, mai puțin cele aflate pe diagonala principală și pe diagonala secundară
b) Suma tuturor elementelor aflate pe poziții cu produsul indicilor par, excluzând cele de pe cele două diagonale principale
c) Suma tuturor elementelor aflate pe poziții cu suma indicilor impară, excluzând cele de pe diagonala secundară
d) Suma tuturor elementelor aflate pe poziții cu suma indicilor pară, incluzând cele de pe diagonala principală

16. Indicați varianta de răspuns corectă ce conține declararea vectorului v și a valorii lui n pentru care apelul funcției f(v, n) întoarce valoarea 1.

```
int f(int v[], int n) {
      int i;
      if (n < 2) return 1;

      for (i = 1; i < n; i++)
            if ((v[i] % 2) == (v[i - 1] % 2))
                  return 0;

      return 1;
}
```

a) `int v[] = { 48, 61, 42, 86, 35, 74 }, n = 6;`
b) `int v[] = { 45, 66, 33, 16, 7, 19 }, n = 5;`
c) `int v[] = { 25, 66, 33, 17 }, n = 4;`
d) `int v[] = { 15, 12, 73, 14, 22 }, n = 5;`

17. Se consideră subprogramul de mai jos. Care este rezultatul apelului acestuia? Se consideră că numerele din vectorul v sunt mereu pozitive, iar *n* este numărul de elemente.

```
void print(int v[], int n){
      int i, k = v[0];
      if (n == 1){
            printf("%d", k);
            return;
      }
      for (i = 1; i < n; i++){
            int a = k;
            int b = v[i];
            while (a != b)
                  if (a > b)
                        a -= b;
                  else
                        b -= a;
            k = a;
      }
      printf("%d", k);
}
```

a) Cel mai mic multiplu comun al elementelor vectorului
b) Cel mai mare divizor comun al elementelor vectorului
c) Cel mai mare număr prim din vector
d) Cel mai mare divizor comun prim al elementelor vectorului

18. Se consideră programul de mai jos. Specificați pentru ce valoare a numărului întreg *N* se va afișa pe ecran șirul *„Corect!”*.

```
void main() {
      int sum=0,i=0,N = . . . . . ;
      while (N > 0) {
            int digit = N % 10;
            if (digit % 2 == 0 && i % 2 != 0)
                  sum += digit;
            if (digit % 2 != 0 && i % 2 == 0)
                  sum += digit;
            N = N / 10;
            i++;
      }
      if (sum == 10)
            printf("Corect!");
      else
            printf("Incorect!");
}
```

a) 14056932 **b)** 123456 **c)** 90018 **d)** 23965041

19. Se consideră subprogramul de mai jos, unde *x* reprezintă un număr natural nenul. Ce valoare va returna apelul funcției *test*(*30112904*)?

```
int test(int x){
      int r = 0;
      while (x != 0)
      {
            if(x % 2 != 0)
                  r = r * 10 + x % 10;
            x /= 10;
      }
      return r;
}
```

a) 91103 **b)** 3119 **c)** 301190 **d)** 9133

20. Ce afișează programul de mai jos?

```
void main(){
      int a = 10031, b = 20194510;
      int c = 0, aux, c_a, c_b, i, f = 0, z = 1, r;
      while (b > 0) {
            c_b = b % 10;
            aux = a;
            f = 0;
            while (aux != 0)
            {
                  c_a = aux % 10;
                  if (c_a == c_b)
                  {
                        f = 1;
                        break;
                  }
                  aux /= 10;
            }
            if (f == 0)
            {
                  r = c_b % 2 + 1;
                  for (i = 0; i < r; i++) {
                        c = c + z * c_b;
                        z = z * 10;
                  }
            }
            b /= 10;
      }
      printf("%d", c);
}
```

a) 29455 **b)** 2994555 **c)** 299455 **d)** 294455

21. Ce afișează programul de mai jos?

```
void main() {
	int i = 0, k = 0, n = 533657;
	while (n)
	{
		i = i * 10 + n % 10;
		if (i % 2 == 0)
			break;
		n = n / 100;
		k++;
	}
	printf("%d", i);
}
```

a) 76 **b)** 635 **c)** 763 **d)** 56

22. Considerând că funcția *f(int x)* returnează numărul de cifre pare din numărul *x*, care din variantele următoarele poate fi introdusă ca date de intrare pentru secvența de program următoare astfel încât aceasta să afișeze *"DA"*?

```
int a[100], c[100], n;
int nr1, nr2, i, j = 0;

scanf("%d", &n);
for (i = 0; i < n; i++)
	scanf("%d", &a[i]);

for (i = 0; i < n / 2; i++)
{
	c[j] = f(a[i]);
	c[n - 1 - j] = f(a[n - 1 - i]);
	j++;
}
for (i = 0; i < n / 2; i++)
	if (c[i] != c[n - 1 - i])
		break;

if (i < n / 2)
	printf("NU");
else
	printf("DA");
```

a) 6 122 10 3 5 78 890
b) 6 123 10 3 5 78 80
c) 6 1 3 3 45 5 6
d) 6 203 12 5 8 261 215

23. Ce afișează programul de mai jos?

```
void main(){
	int my_value = 0;
	int a = 2527190;
	while (a)
	{
		my_value += a % 10;

		if (!(a / 10))
		{
			if ((my_value / 10) % 10)
			{
				a = my_value;
				my_value -= my_value;
			}
			else
				a /= 10;
		}
		else
			a /= 10;
	}
	printf("%d", my_value);
}
```

a) 4 **b)** 2 **c)** 3 **d)** 8

24. Ce reprezintă valoarea întoarsă de funcția *func* definită mai jos?

```
int func(){
	int f1 = 1, f2 = 1, fn;
	int n = 1547, val = 0, i, counter;
	while (1){
		counter = 0;
		fn = f1 + f2;
		if (fn >= n)
			break;
		for (i = 1; i <= fn; i++)
			if (!(fn % i))
				++counter;
		if (counter <= 2)
			val++;
		f1 = f2;
		f2 = fn;
	}
	return val;
}
```

a) Numărul de elemente din șirul lui Fibonacci, strict mai mici decât 1547
b) Numărul de elemente din șirul lui Fibonacci care au cel puțin 3 divizori
c) Suma elementelor din șirul lui Fibonacci, strict mai mici decât 1547 și strict mai mari decât 2
d) Numărul de elemente prime din șirul lui Fibonacci, strict mai mici decât 1547

25. Precizați câte perechi de numere va afișa apelul *f(v, 8)* pentru funcția *f* și vectorul v definite mai jos.

```
int v[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
void f(int v[], int n){
      int i, j;
      for (i = 0; i < n; i++)
            for (j = i + 1; j < n; j++)
                  if (v[i] < v[j])
                        if(v[i] % 2 + v[j] % 3 == 0)
                              printf("(%d,%d)", v[i], v[j]);
}
```

a) 0 **b)** 2 **c)** 1 **d)** 3

26. Dacă n este o variabilă întreagă având valoarea 16, precizați ce valoare are variabila *sum* după execuția secvenței de cod următoare:

```
int i, j, k, sum = 0;
for (i = 1; i < n; i *= 2)
      for (j = n; j > 0; j /= 2)
            for (k = j; k < n; k += 2)
                  sum++;
```

a) 68 **b)** 100 **c)** 140 **d)** 3325

27. Se consideră subprogramul de mai jos. Ce se va afișa la apelul funcției pentru valoarea *n = 5* și *v = {123, 122, 244, 135, 9}* ?

```
void print(int v[], int n){
      int i, s = 0;
      bool impar, flag;
      for (i = 0; i < n; i++)
      {
            impar = v[i] % 2;
            flag = true;
            while (v[i] > 9)
            {
                  v[i] /= 10;
                  if (impar == v[i] % 2)
                  {
                        flag = false;
                        break;
                  }
                  impar = v[i] % 2;
            }
            if (flag)
                  s++;
      }
      printf("%d", s);
}
```

a) 1 **b)** 3 **c)** 0 **d)** 2

28. Se consideră subprogramul de mai jos, unde *matrice* reprezintă o matrice pătratică de numere întregi, populată cu valori de *0*, iar *N* numărul de rânduri, respectiv coloane ale matricei. Ce valoare va avea elementul $matrice\left[\frac{N}{2}\right]\left[\frac{N}{2}\right]$, în urma apelului funcției *f(matrice, 9)*?

```
void f(int matrice[9][9], int N){
      int i, j;
      for (i = 0; i <= N / 2; i++)
            for (j = i; j < N - i; j++)
                  if (i == 0)
                        matrice[i][j] = 1;
                  else
                        matrice[i][j]=matrice[i-1][j-1]+
                  matrice[i-1][j] + matrice[i-1][j+1];
}
```

a) $2^{N/2}$ **b)** $2^{N/2+1}$ **c)** $3^{N/2-1}$ **d)** $3^{N/2}$

29. Se consideră subprogramul *print* de mai jos. Ce se va afișa la apelul funcției pentru valoarea *matrix* de mai jos?

$$matrix = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 5 & 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

```
void print(int matrix[5][5]) {
      int n = 5;
      int top=0, bottom = n-1, left=0, right = n-1;
      while (top <= bottom && left <= right)
      {
            for (int i = left; i <= right; i++)
                  printf("%d ", matrix[top][i]);
            top++;
            for (int i = top; i <= bottom; i++)
                  printf("%d ", matrix[i][right]);
            right--;
            if (top <= bottom)
            {
                  for (int i = right; i >= left; i--)
                        printf("%d ", matrix[bottom][i]);
                  bottom--;
            }
            if (left <= right)
            {
                  for (int i = bottom; i >= top; i--)
                        printf("%d ", matrix[i][left]);
                  left++;
            }
      }
}
```

a) 1 2 3 4 5 1 0 1 5 4 3 2 1 5 0 5 4 3 2 0 2 3 4 0 0
b) 0 0 4 3 2 0 2 3 4 5 0 5 1 2 3 4 5 1 0 1 5 4 3 2 1
c) 1 5 0 5 1 2 3 4 5 1 0 1 5 4 3 2 4 0 4 3 2 0 2 3 0
d) 1 3 5 1 2 3 4 0 1 2 3 1 4 2 3 5 1 2 3 1 4 1 2 3 1

30. Se consideră funcția *parcurgere* și matricea *matrice* de mai jos. *N* este numărul de linii și de coloane ale matricei. Menționați care este secvența afișată de apelul *parcurgere(matrice,4)*.

$$matrice = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

```
void parcurgere(int matrice[4][4], int N){
      int i, j, start = 0;
      while (start < 2 * N)
      {
            int limita = start;
            if (limita >= N)
                  limita = N - 1;

            for (j = limita; j >= 0; j--)
                  for (i = 0; i <= limita; i++)
                        if (i + j == start)
                              printf("%d ", matrice[i][j]);
            start++;
      }
}
```

a) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
b) 1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16
c) 1 4 16 13 5 8 12 9 6 7 11 10 2 3 15 14
d) 1 2 5 3 6 9 4 7 10 13 8 11 14 12 15 16

31. Se consideră subprogramul de mai jos, unde *v* și *u* reprezintă doi vectori de elemente întregi, iar *n* numărul de elemente ai vectorilor *v și u*. Menționați valorile vectorilor după apelul funcției *modify*, știind că inițial vectorii au valorile $v = \{4,0,1,3,2\}$ și $u = \{2,1,0,3,4\}$.

```
void modify(int v[], int u[], int n){
      int i;
      for (i = 0; i < n; i++)    {
            v[i] = u[v[i]];
            u[i] = v[u[i]];
      }

}
```

a) $v = \{1,1,4,3,4\}$ și $u = \{4,1,1,3,4\}$
b) $v = \{4,1,1,3,4\}$ și $u = \{1,1,4,3,4\}$
c) $v = \{4,1,1,3,4\}$ și $u = \{4,0,1,3,2\}$
d) $v = \{2,1,0,3,4\}$ și $u = \{4,0,1,3,2\}$

32. Se consideră subprogramul de mai jos. Ce afișează acesta?

```
void print(){
      int m[4][4] = { {1, 2, 3, 4},
                      {5, 6, 7, 8},
                      {9,10,11,12},
                      {13,14,15,16 } };
      int i, j ,n = 4;
      for (i = 0; i < n; i++, printf("\n"))
            for (j = 0; j < n; j++)
                  printf("%d ", m[i][(i + j) % n]);
}
```

a) $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$

b) $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 12 & 11 & 10 & 9 \\ 16 & 15 & 14 & 13 \end{pmatrix}$

c) $\begin{pmatrix} 4 & 3 & 2 & 1 \\ 8 & 7 & 6 & 5 \\ 12 & 11 & 10 & 9 \\ 16 & 15 & 14 & 13 \end{pmatrix}$

d) $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 6 & 7 & 8 & 5 \\ 11 & 12 & 9 & 10 \\ 16 & 13 & 14 & 15 \end{pmatrix}$

33. Se consideră subprogramul de mai jos, unde *v și u* reprezintă doi vectori de elemente întregi, iar *n* numărul de elemente ai vectorilor *v și u*. Menționați ce va afișa apelul funcției *print* pentru vectorii $v = \{4, 0, 1, 3, 2\}$ și $u = \{3, 5, 7, 9, 11\}$.

```
void print(int v[], int u[], int n){
      int i;
      for (i = 0; i < n; i++)
            if (i % 2)
                  printf("%d", u[v[i]]);
            else
                  printf("%d", u[v[i] - 1]);
}
```

a) 9 3 3 9 5 **b)** 3 9 9 3 11 **c)** 9 5 5 9 11 **d)** 9 7 7 5 3

34. Se consideră subprogramul de mai jos, unde *m* reprezintă o matrice pătratică de elemente întregi, iar *n* numărul de linii și de coloane ale lui *m*. Menționați valoarea matricei *m* după apelul funcției *print*, dacă valoare ei inițială este:

$$m = \begin{pmatrix} 0 & 2 & 3 & 1 \\ 2 & 0 & 1 & 3 \\ 3 & 1 & 0 & 2 \\ 1 & 3 & 2 & 0 \end{pmatrix}$$

```
void print(int m[4][4], int n){
      int i, j;
      for (i = 0; i < n; i++)
      {
            for (j = 0; j < i; j++)
            {
                  m[i][j] = n * m[i][j];
                  m[j][i] = n * m[j][i];
            }
      }
}
```

a) $\begin{pmatrix} 0 & 2 & 3 & 1 \\ 2 & 0 & 1 & 3 \\ 3 & 1 & 0 & 2 \\ 1 & 3 & 2 & 0 \end{pmatrix}$ **b)** $\begin{pmatrix} 0 & 8 & 12 & 4 \\ 2 & 0 & 4 & 12 \\ 3 & 1 & 0 & 8 \\ 1 & 3 & 2 & 0 \end{pmatrix}$ **c)** $\begin{pmatrix} 0 & 8 & 12 & 4 \\ 8 & 0 & 4 & 12 \\ 12 & 4 & 0 & 8 \\ 4 & 12 & 8 & 0 \end{pmatrix}$ **d)** $\begin{pmatrix} 0 & 8 & 12 & 4 \\ 2 & 0 & 4 & 12 \\ 12 & 4 & 0 & 2 \\ 4 & 12 & 8 & 0 \end{pmatrix}$

35. Precizați dintre afirmațiile de mai jos este corect verificată de funcția următoare (afișează mesajul *result: true*) când aceasta este aplicată asupra unei matrice pătratice oarecare *A* de dimensiune *n*, cu $n \leq 100$.

```
void check_matrix(int A[ ][100], int  n){
      int i, j, k, s = 0;
      for (j = 0; j < n; j++)
            for (i = 0; i < j; i++)
                  if (A[i][j] > A[j][i])
                        s++;
      if (s >= (n * (n - 1) / 2))
            printf("result: true\n");
      else
            printf("result: undefined\n");
}
```

a) matricea conține cel puțin $\frac{n(n-1)}{2}$ elemente diferite între ele

b) suma tuturor elementelor aflate deasupra diagonalei principale este mai mare decât suma tuturor elementelor aflate sub diagonala principală

c) suma tuturor elementelor aflate sub diagonala principală este mai mare decât suma tuturor elementelor aflate deasupra diagonalei principale

d) matricea conține valorile șirului $1, 2, 3, 4, \ldots\ldots, n-1$

36. Se consideră matricea *A*:

$$A = \begin{pmatrix} x & y & y & x \\ y & x & x & y \\ y & x & x & y \\ x & y & y & x \end{pmatrix}$$

Alegeți o variantă de răspuns care să completeze spațiile punctate astfel încât matricea *m* să fie egală cu matricea *A*.

```
char m[4][4];
int i = 0, j = 0;
int n = 4;
for (i = 0; i < n; i++)
{
      for (j = 0; j < n; j++)
      {
            . . . . . .
      }
}
```

a)
```
if ((i == j) || (i + j == n - 1))
      m[i][j] = 'y';
else
      m[i][j] = 'x';
```

b)
```
if ((i == j) || !(i + j == n - 1))
      m[i][j] = 'y';
else
      m[i][j] = 'x';
```

c)
```
if ((i == j) && !(i + j == n - 1))
      m[i][j] = 'x';
else
      m[i][j] = 'y';
```

d)
```
if ((i != j) && !(i + j == n - 1))
      m[i][j] = 'y';
else
      m[i][j] = 'x';
```

37. Se consideră secvența de instrucțiuni de mai jos:

```
int i, j, k;
int a[3][3] = . . . . . .;
int b[3][2];
int c[3][2] = { 0 };

for (i = 0; i < 3; i++)
      for (j = 0; j < 2; j++)
            b[i][j] = i + j;

for (i = 0; i < 3; i++)
      for (j = 0; j < 2; j++)
            for (k = 0; k < 3; k++)
                  c[i][j] += a[i][k] * b[k][j];
```

Cum trebuie completat spațiul punctat, astfel încât la finalul execuției, matricea *c* să conțină valorile $\{\{0,1\},\{1,2\},\{2,3\}\}$?

a) $\{\{0,0,0\},\{0,0,0\},\{0,0,0\}\}$
b) $\{\{1,1,1\},\{1,1,1\},\{1,1,1\}\}$
c) $\{\{1,0,0\},\{0,1,0\},\{0,0,1\}\}$
d) $\{\{0,0,1\},\{0,1,0\},\{1,0,0\}\}$

38. Precizați care sunt elementele matricei *m* după execuția funcției *f*:

```
void f(char m[4][4]){
      int i, j;
      for (i = 0; i < 4; i++)
      {
            for (j = 0; j < 4; j++)
            {
                  if (!((i + j) % 2) && (i + j) < 3)
                        m[i][j] = 'a';
                  else if (((i + j) % 2) && (i + j) > 3)
                        m[i][j] = 'b';
                  else
                        m[i][j] = '1';
            }
      }
}
```

a) $\begin{pmatrix} a & a & a & a \\ b & a & a & a \\ b & b & a & a \\ b & b & b & a \end{pmatrix}$

b) $\begin{pmatrix} 1 & a & a & a \\ b & 1 & a & a \\ b & b & 1 & a \\ b & b & b & 1 \end{pmatrix}$

c) $\begin{pmatrix} a & 1 & a & 1 \\ 1 & a & 1 & 1 \\ a & 1 & 1 & b \\ 1 & 1 & b & 1 \end{pmatrix}$

d) $\begin{pmatrix} 1 & 1 & a & 1 \\ b & 1 & 1 & a \\ 1 & b & 1 & 1 \\ b & 1 & b & 1 \end{pmatrix}$

39. Considerăm matricea $A = \begin{pmatrix} 3 & 5 \\ 2 & 7 \end{pmatrix}$. Numerotarea elementelor începe de la 1 (ex: $A[1][2] = 5$). Matricea B este o matrice $2x2$ cu toate elementele nule inițial. Indicați valorile matricei B în urma execuției următoarei secvențe de cod.

```
for (i = 1; i <= n; i++){
      for (j = 1; j <= n; j++){
            a = (i == 1 ? 2 : 1);
            b = (j == 1 ? 2 : 1);
            B[i][j] = ((i + j) % 2 != 0) ? -1 : 1;
            B[i][j] *= A[a][b];
      }
}
```

a) $\begin{pmatrix} 3 & -5 \\ -2 & 7 \end{pmatrix}$ **b)** $\begin{pmatrix} 4 & 9 \\ 3 & 2 \end{pmatrix}$ **c)** $\begin{pmatrix} 7 & 22 \\ 43 & 9 \end{pmatrix}$ **d)** $\begin{pmatrix} 7 & -2 \\ -5 & 3 \end{pmatrix}$

40. Indicați ce afișează programul de mai jos.

```
int f(int n){
      int d = 2, p, pm=0;
      while (n > 1)
      {
            if (n % d == 0)
            {
                  p = 0;
                  while (n % d == 0)
                  {
                        ++p;
                        n /= d;
                  }
                  if (pm < p)
                        pm = p;
            }
            ++d;
      }
      return pm;
}
void main(){
    int v[] = { 2048, 4096, 6561 };
    int i, s = 0;
    for (i = 0; i < 3; i++)
        s += f(v[i]);
    printf("%d", s);
}
```

a) 31 **b)** 29 **c)** 42 **d)** 30

41. Indicați ce afișează programul de mai jos:

```
void main()
{
	int m[4][4] = {
	{22,   13,  4, 121},
	{32,   44, 18,  23},
	{22,  242, 44,  33},
	{11, 1221,  6,  14} };

	int i, j, s = 0;
	for (i = 0; i < 4; i++)
	{
		if (i % 2 == 1)
		{
			for (j = 0; j < 4; j++)
			{
				int n, c, r;
				n = m[i][j];
				c = n;
				r = 0;
				while (n > 0)
				{
					r = r * 10 + n % 10;
					n /= 10;
				}
				if (c == r && c % 2 == 0)
					s += c;
			}
		}
	}
	printf("%d", s);
}
```

a) 1271 **b)** 44 **c)** 50 **d)** 121

42. Indicați valorile matricei *v*, după execuția următoarei secvențe:

```
int n = 3, m = 5;
int i, j, aux;
int v[3][5]={{0,1,2,3,4},{5,6,7,8,9},{10,11,12,13,14}};
for (i = 0; i < n; i++) {
	for (j = 0; j < m / 2; j++) {
		aux = v[i][j];
		v[i][j] = v[i][m - j - 1];
		v[i][m - j - 1] = aux;
	}
}
```

a) $\begin{pmatrix} 4 & 3 & 2 & 1 & 0 \\ 9 & 8 & 7 & 6 & 5 \\ 14 & 13 & 12 & 11 & 10 \end{pmatrix}$

c) $\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 9 & 8 & 7 & 6 & 5 \\ 10 & 11 & 12 & 13 & 14 \end{pmatrix}$

b) $\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 & 9 \\ 10 & 11 & 12 & 13 & 14 \end{pmatrix}$

d) $\begin{pmatrix} 14 & 13 & 12 & 11 & 10 \\ 9 & 8 & 7 & 6 & 5 \\ 4 & 3 & 2 & 1 & 0 \end{pmatrix}$

43. Fie o matrice numită *pasi* de dimensiune *4x12*. Aceasta stochează pe fiecare dintre linii pașii făcuți de o persoană în fiecare lună de-a lungul unui an. Matricea are 4 linii, corespunzătoare anilor *2021-2024*. Menționați cu ce trebuie înlocuită linia punctată astfel încât codul de mai jos să afișeze numărul maxim de pași înregistrat în *3* luni consecutive din intervalul *2021-2024*.

```
int main() {
      int pasi[4][12];
      int sum = 0;
      int max_sum = 0;
      int old_value = 0;

      for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 12; j++) {
                  if (i == 0 && j < 3) {
                        max_sum = sum + pasi[i][j];
                        sum = max_sum;
                  }
                  else
                  {
                        if (j - 3 < 0)
                              old_value = . . . . . .
                        else
                              old_value = pasi[i][j - 3];
                        sum = sum + pasi[i][j] - old_value;
                        if (sum > max_sum) {
                              max_sum = sum;
                        }
                  }
            }
      }
      printf("%d\n", max_sum);
}
```

a) `pasi[i][(j - 2)];`
b) `pasi[i][(j + 12 - 3) % 12];`
c) `pasi[i - 1][j + 12 - 3];`
d) `pasi[i-1][(j - 2)];`

44. Pornind de la o matrice *X* cu *M* linii și *M* coloane, unde *M* este un număr par mai mare decât *0*, ne propunem să creăm o matrice *Y* de *M/2* linii și *M/2* coloane. Aceasta va conține pe fiecare poziție suma elementelor sub-matricelor $2x2$ din matricea inițială. Care dintre următoarele secvențe de cod rezolvă această cerință? Exemplu:

$$\begin{pmatrix} 1 & 2 & 4 & 2 \\ 3 & 2 & 1 & 1 \\ 3 & 4 & 1 & 0 \\ 2 & 2 & 2 & 1 \end{pmatrix} => \begin{pmatrix} 8 & 8 \\ 11 & 4 \end{pmatrix}$$

a)
```
for (int i = 0; i < M; i += 2)
    for (int j = 0; j < M; j += 2)
        Y[i/2][j/2]=X[i][j]+X[i+1][j]+X[i][j+1]+X[i+1][j+1];
```

b)
```
for (int i = 0; i < M/2; i++)
    for (int j = 0; j < M/2; j++)
        Y[i/2][j/2]=X[i][j]+X[i+1][j]+X[i][j+1]+X[i+1][j+1];
```

c)
```
for (int i = 0; i < M/2; i++)
    for (int j = 0; j < M/2; j++)
        Y[i][j]=X[i][j]+X[i+1][j]+X[i][j+1]+X[i+1][j+1];
```

d)
```
for (int i = 0; i < M/2; i++)
    for (int j = 0; j < M/2; j++)
        Y[i/2][j/2]=X[i][j]+X[i+1][j+1];
```

45. Pentru funcția f definită mai jos și valorile *n* și *m* date, cu $1 \leq n \leq 100$ și $8 \leq m \leq 100$, apelul *f(n, m)* va afișa valoarea *406* pe coloana a cincea de pe o anumită linie. Să se precizeze pe ce linie și pe ce coloană afișează același apel al funcției *f* valoarea *872*. Se consideră că indexarea liniilor și coloanelor se face începând cu *0*.

```
void f(int n, int m){
    int i, j, value = 0;
    for (i = 0; i < n; i++)
        for (j = 0; j < m; j++)
            printf("%d ", value++);
        printf("\n");
}
```

a) linia 5, coloana 10
b) linia 9, coloana 3
c) linia 13, coloana 1
d) linia 12, coloana 7

46. Precizați ce efect are secvența de cod de mai jos aplicată unui vector *v* ce conține *n* numere întregi de câte *6* cifre.

```
int k, i;
for (k = 0; k < n; )
{
      if (v[k] / 100000 != v[k] % 10 ||
            v[k] / 10000 % 10 != v[k] / 10 % 10 ||
            v[k] / 1000 % 10 != v[k] / 100 % 10)
            {
                  i = k + 1;
                  while (i < n)
                        {
                              v[i - 1] = v[i];
                        i++;
                        }
                  n--;
            }
      else
            k++;
}
```

a) elimină elementele din vector care sunt divizibile cu 10
b) elimină elementele din vector care nu sunt divizibile cu 10
c) elimină elementele din vector care nu sunt palindrom
d) sortează crescător elementele din vector, după numărul de cifre de 0

47. Se consideră programul de mai jos. Ce se afișează pe ecran?

```
void main() {
      int i, j, k, l;
      int v2[2][1] = { 0 };
      int v1[2][1] = { 1,1 };
      int mat[2][2] = { 0,-1,1,0 };
      for (l = 0; l < 5; l++){
            for (int i = 0; i < 2; i++)
                  for (j = 0; j < 1; j++)
                        for (k = 0; k < 2; k++)
                              v2[i][j]+=mat[i][k]*v1[k][j];
            v1[0][0] = v2[0][0];
            v1[1][0] = v2[1][0];
            v2[0][0] = 0;
            v2[1][0] = 0;
      }
      printf("%d %d\n", v1[0][0], v1[1][0]);
}
```

a) -1 1 **b)** 1 1 **c)** 1 -1 **d)** -1 -1

48. Se consideră programul de mai jos. Ce afișează acesta pe ecran?

```
int main()
{
	int i, j, k, l;
	int v2[2] = { 0 };
	int v1[2] = { 1,1 };
	int mat[2][2] = { 0,-1,1,0 };

	for (l = 0; l < 1022; l++)
	{
		for (j = 0; j < 2; j++)
		{
			for (k = 0; k < 2; k++)
				v2[j] += mat[j][k] * v1[k];
		}
		v1[0] = v2[0];
		v1[1] = v2[1];
		v2[0] = 0;
		v2[1] = 0;
	}
	printf("%d %d\n", v1[0], v1[1]);
	return 0;
}
```

a) -1 1 **b)** -1 -1 **c)** 1 1 **d)** 2 -2

49. Ce afișează programul de mai jos?

```
void f(int v[], int n, int k)
{
	int temp[10];
	for (int i = 0; i < n; i++)
		temp[(i + k) % n] = v[i];

	for (int i = 0; i < n; i++)
		v[i] = temp[i];
}
void main()
{
	int v[] = { 1, 2, 3, 4, 5 };
	f(v, 5, 2027);

	for (int i = 0; i < 5; i++)
		printf("%d ", v[i]);
}
```

a) 4 5 1 2 3 **b)** 5 4 3 2 1 **c)** 3 4 5 1 2 **d)** 2 3 4 5 1

50. În secvența de instrucțiuni de mai jos variabilele y, *l* și c sunt două variabile întregi, iar variabila *M* reprezintă un tablou bidimensional cu *5* linii și *5* coloane. Se consideră de asemenea secvența de instrucțiuni de mai jos. Indicați care este varianta corectă de utilizat în locul spațiului punctat astfel matricea *A* să conțină elementele din figura de mai jos:

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

```
for (int y = 0; y < 5; y++)
      for (int l = y; l < 5 - y; l++)
            for (int c = y; c < 5 - y; c++)
                  . . . . . .
```

a) *M[l][c] = 5 - y;*
b) *M[l][c] = i + j + 1;*
c) *M[l][c] = y + 1;*
d) *M[l][c] = i + j - 1;*

51. În secvența de instrucțiuni de mai jos variabilele *l* și c sunt două variabile întregi, iar variabila *M* reprezintă un tablou bidimensional cu *5* linii și *5* coloane. Indicați care este varianta corectă de utilizat în locul spațiului punctat astfel după execuția secvenței de cod de mai jos, matricea *A* să conțină următoarele elemente:

$$A = \begin{pmatrix} x & z & y & x & x \\ z & y & x & x & z \\ y & x & x & z & y \\ x & x & z & y & x \\ x & z & y & x & x \end{pmatrix}$$

```
for (l = 0; l < 5; l++)
      for (c = 0; c < 5; c++)
            . . . . . .
                  M[l][c] = 'x';
            else
                  if ((l + c) % 4 == 1)
                        M[l][c] = 'z';
                  else
                        if ((l + c) % 4 == 2)
                              M[l][c] = 'y';
```

a) *if ((l + c) % 4 == 0 || (l + c) % 4 < 4)*
b) *if ((l + c) % 4 == 0)*
c) *if ((l + c) % 4 == 3)*
d) *if ((l + c) % 4 == 0 || (l + c) % 4 == 3)*

52. Indicați ce afișează execuția programului următor:

```
void main() {
      int i = 0, c = 3;
      char s[50] = "PROGRAMAREINLIMBAJULC";
      int len = strlen(s);
      for (i = 0; i < len ; i += 1) {
            char a = s[i];
            s[i] = s[len - 1 - i];
            s[len - 1 - i] = a;
      }
      printf("%s", s);
}
```

a) PROGRAMAREINLIMBAJULC
b) CLUJABMILNIERAMARGORP
c) CJARGMRINLIAAMBAORPLU
d) CJARGMRINLIAAMBAOPPLU

53. Care este conținutul matricei A după execuția următorului program?

```
void main(){
      int A[24][24], n=4, i, j, k = 1, s;

      for (i = 0; i < n; i++) {
            k = i + 1;
            for (j = 0; j < n; j++) {
                  if (i % 2 == 0)
                        A[j][i] = k;
                  else
                        A[n - j - 1][i] = k;
                  k *= 10;
            }
      }
}
```

a) $\begin{pmatrix} 1 & 2000 & 3 & 4000 \\ 10 & 200 & 30 & 400 \\ 100 & 20 & 300 & 40 \\ 1000 & 2 & 3000 & 4 \end{pmatrix}$ **b)** $\begin{pmatrix} 1 & 10 & 100 & 1000 \\ 2 & 20 & 200 & 2000 \\ 3 & 30 & 300 & 3000 \\ 4 & 40 & 400 & 4000 \end{pmatrix}$

c) $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 10 & 20 & 30 & 40 \\ 100 & 200 & 300 & 400 \\ 1000 & 2000 & 3000 & 4000 \end{pmatrix}$ **d)** $\begin{pmatrix} 1 & 10 & 100 & 1000 \\ 2000 & 200 & 20 & 2 \\ 3 & 30 & 300 & 3000 \\ 4000 & 400 & 40 & 4 \end{pmatrix}$

54. Ce realizează funcția *f* definită mai jos, având ca parametri o matrice pătratică de *nxn* întregi în care indexarea începe de la *1*?

```
int f(int m[100][100], int n){
      int a, b, nr = 0, ok;
      int vx[] = { 0,1,0,-1 };
      int vy[] = { 1,0,-1,0 };
      for (int i = 1; i <= n; i++)
      {
            for (int j = 1; j <= n; j++)
            {
                  ok = 1;
                  for (int t = 0; t < 4; t++)
                  {
                        a = i + vx[t];
                        b = j + vy[t];
                        if (a>=1 && a <= n && b>=1 && b<=n)
                        {
                              if (m[i][j] > m[a][b])
                              {
                                    ok = 0;
                                    break;
                              }
                        }
                  }
                  if (ok == 1)
                        nr++;
            }
      }
      return nr;
}
```

a) Funcția determină numărul de elemente din matrice care sunt maxime în raport cu vecinii lor.

b) Funcția determină numărul de elemente din matrice care sunt minime în raport cu vecinii lor.

c) Funcția determină numărul elementelor care au 4 vecini din matrice.

d) Funcția determină dacă există cel puțin un element care să fie maxim în raport cu vecinii săi.

55. Indicați valorile elementelelor matricei *mat* după execuția funcției *my_func*, definită mai jos.

```
int mat[3][3] = { {1,2,3} , {4,5,6}, {7,8,9} };
void my_func(){
      int n = 3, i, j, val;
      for (i = 0; i < n; i++)
      {
            val = mat[i][0];
            for (j = 0; j < n; j++)
                  if (val < mat[i][j])
                        val = mat[i][j];
            mat[i][n - 1 - i] = val;
      }
}
```

a) $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ **b)** $\begin{pmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{pmatrix}$ **c)** $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 6 & 6 \\ 9 & 8 & 9 \end{pmatrix}$ **d)** $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 6 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

56. Precizați ce valoare întoarce apelul *f(5,5,7)* pentru funcția *f* definită mai jos.

```
int f(int m, int n, int offset) {
      int i, j, nivel, mat[10][10] = { 0 }, t=0, s=0, min;
      min = (m < n) ? m : n;
      for (nivel = 0; nivel < min / 2; nivel++)
      {
            t++;
            for (i = nivel; i < m - nivel; i++)
            {
                  t--;
                  for (j = nivel; j < n - nivel; j++)
                        mat[i][j] = nivel + offset + t;
            }
      }
      for (i = 1; i < n; i++)
            for (j = 0; j < i; j++)
                  s += mat[i][j];
      return s;
}
```

a) 14 **b)** 21 **c)** 34 **d)** 86

ȘIRURI DE CARACTERE

57. Se consideră secvența de cod de mai jos, unde *s1*, *s2* și *s3* reprezintă șiruri de caractere. Menționați ce se va afișa pe ecran, în urma executării programului.

```
char s1[] = "Academia Tehnica Militara Ferdiand I";
char s2[] = "Admitere2025";
char s3[100] = "Student";
strcpy(s1, s1 + 17);
strcpy(s1 + 7, s1 + strlen(s1));
strcpy(s2, s2 + 8);
strcat(s1, s2);
strcat(s3, s1);
printf("%s", s3);
```

a) Student2025
b) AcademieMilitar2025
c) StudentMilitar2025
d) AdmitereMilitar2025

58. Indicați ce afișează programul următor:

```
void main()
{
	char cuvant[20] = "AdmitereMTA2024";
	int nr = 4, i=0;
	while (i < nr)
	{
		strcpy(cuvant, cuvant + i);
		i++;
	}
	char nou[20];
	strcpy(nou, cuvant + 2);
	nou[strlen(nou) % (strlen(cuvant)+1) - 1] += 1;
	nou[0] += ('b' - 'B');
	printf("%s", nou);
}
```

a) *ereMTA2025* **b)** *eMTA2034* **c)** *MTA2025* **d)** *mTA2025*

59. Precizați care este șirul de caractere ce va fi afișat în urma execuției programului următor:

```
void main() {
      char sir[100] = "Acesta este un exercitiu!";
      int i, j, aux = 0;
      for (i = 0; i < strlen(sir); i++) {
            if (i * i == strlen(sir)) {
                  aux = i;
                  break;
            }
      }
      for (i = 0; i < aux; i++)
            printf("%c", sir[i * aux + (aux - i - 1)]);
}
```

a) A ur! **b)** Acest **c)** itiu! **d)** tsuxi

60. Menționați ce valoare se va regăsi în variabila *s* în urma execuției programului următor:

```
void main() {
      int k = 1;
      char sir[50] = "ana are mere";
      int i;

      for (i = 0; i < strlen(sir); i++) {
            if (sir[i] != ' ')
                  sir[i] += k;

            if (sir[i] == ' ')
                  k += 1;
      }
      printf("%s", sir);
}
```

a) *bob ctg phuh*
b) *cpc duh qivi*
c) *cpc evi skxk*
d) *cpc ctg ogtg*

61. Se consideră următoarea secvență de instrucțiuni. Ce se afișează în urma executării acesteia?

```
char sir1[100] = "academia militara";
char sir2[20] = "ethnica";
char aux, rez[100];

aux = sir2[0];
sir2[0] = sir2[1];
sir2[1] = aux;

char* p = strtok(sir1, " ");
strcpy(rez, p);
strcat(rez, sir2);
p = strtok(NULL, " ");
strcat(rez, p);
rez[strlen(sir2)] -= 32;
printf("%s", rez);
```

a) *AcademiaTehnicaMilitara*
b) *academiaTehnicaMilitara*
c) *academiAtehnicamilitara*
d) *academiatehnicaMilitara*

62. Ce se va afișa în urma executării programului de mai jos?

```
char sir[2025];
strcpy(sir, "2025ACADEMIAtehnicaMILITARA2025");
int i;
for (i = 0; i < strlen(sir); i++)
{
	if (sir[i] >= 'A' && sir[i] <= 'Z')
	{
		if (i > 0)
			if (sir[i-1] >= 'a' && sir[i-1] <= 'z')
				printf(" ");//spatiu
		printf("%c", sir[i]);
		if (sir[i + 1] >= 'a' && sir[i + 1] <= 'z')
			printf(" ");//spatiu
	}
	if (sir[i] >= 'a' && sir[i] <= 'z')
		printf("%c", sir[i] - ('a' - 'A'));
}
```

a) *2025ACADEMIA TEHNICA MILITARA2025*
b) *2025 academia tehnica militara 2025*
c) *academia tehnica militara*
d) *ACADEMIA TEHNICA MILITARA*

63. Se consideră subprogramul de mai jos, unde *s* reprezintă un sir de caractere. Menționați ce va afișa apelul funcției *f*, pentru *s=”academia_tehnica_militara”*.

```
void f(char s[]){
      int i;
      char str[] = "aeiou";
      for (i = 0; i < strlen(s); i++)
      {
            if (strchr(str, s[i]) || s[i] == '_')
            {
                  strcpy(s + i, s + i + 1);
                  i--;
            }
            else
                  s[i] = s[i] – ('c'-'C');
      }
      printf("%s", s);
}
```

a) *CDM THNC MLTR*
b) *ACADEMIA_TEHNICA_MILITARA*
c) *CDMTHNCMLTR*
d) *CDM_THNC_MLTR*

64. Se consideră subprogramul de mai jos, unde *s* reprezintă un șir de maxim *20* de caractere. Menționați ce va afișa apelul funcției *f* pentru *s=„Admitere2025ATM”*.

```
void f(char s[]) {
      for (int i = 0; i < strlen(s); i++)
      {
            strncpy(s + i, s + strlen(s) - 1 - i, 1);
            if (s[i] >= 'A' && s[i] <= 'Z')
                  s[i] += 32;
      }
      s[7] = '\0';
      for (int i = 8; i < strlen(s); i++)
            s[strlen(s) - 1 - i] = s[i / 2];
      for (int i = 0; i < strlen(s) / 2; i++)
            s[strlen(s) - 1 - i] = s[i];

      printf("%s", s);
}
```

a) *mtaAdmitere2025atm*
b) *atm2025*
c) *mta5atm*
d) *MTA2025ATM*

65. Se consideră programul de mai jos. Cu ce trebuie înlocuită linia punctată pentru ca programul să afișeze *admitere2025!* ?

```
int main(){
      char p1[] = "!eh5qm2xi0vi2!", p2[16] = { 0 };
      int c = 12, l = 0, i;
      for (i = 1; i < strlen(p1); i++)
      {
            if (p1[i] >= 'a' && p1[i] <= 'z')
                  p2[l++] = p1[i] - 4;
            else
                  if (p1[i] >= '0' && p1[i] <= '9')
                        ..................
                  else
                        p2[strlen(p2)] = p1[i];
      }
      printf("%s", p2);
}
```

a) `p2[c--] = p1[i];`
b) `p2[--c] = p1[i];`
c) `p2[c--] = p1[i+1];`
d) `p2[--c] = p1[i+1];`

66. Ce se va afișa în urma execuției următorului program?

```
void main()
{
      char str[] = "mango,banana,apple,banana,grape";
      char* p1;
      char* p2;
      int count = 0;
      p1 = strrchr(str, 'b');
      if (p1)
            str[p1 - str] = '\0';

      p2 = strtok(str, ",");

      while (p2)
      {
            count++;
            p2 = strtok(NULL, ",");
      }
      printf("%d", count);
}
```

a) 2 **b)** 4 **c)** 1 **d)** 3

67. Precizați ce afișează programul de mai jos.

```
void func(char a[])
{
	int i;

	for (i=0; i<strlen(a); i++)
		if (!(i%2 != 0 || a[i]<'A' || a[i]>'Z'))
			a[i]=a[i]+32;
		else if (!(i%2 == 0 || a[i]<'a' || a[i]>'z'))
			a[i]='#';
		else
			strcpy(a+i, a+i+1);

	printf("%s", a);
}
void main()
{
	char a[] = "AdMiTeRe_INFO_atm";
	func(a);
}
```

a) *A#M#T#R#I#F#a#m*
b) *a#m#t#r#_info_###*
c) *A#M#T#R#_I#F#_a#m*
d) *a#m#t#r#IFoam*

68. Se consideră funcția *f* de mai jos. De asemenea, considerăm caracterul *'a'* ca fiind impar, *'b'* par, *'c'* impar, șamd. Ce va returna funcția *f* pentru *sir="informatica"* și *i=0*?

```
void f(char sir[], int i)
{
	if (i >= strlen(sir))
		return;

	if (sir[i] % 2 == 0)
	{
		strcpy(sir + i, sir + i + 1);
		f(sir, i);
	}
	else
		f(sir, i + 1);
}
```

a) iomaica **b)** iomaca **c)** iomaic **d)** inform

69. Menționați ce valoare va avea *s* după execuția secvenței de mai jos:

```
char s[] = "ADMitere";
int i;

for (i = 0; i < strlen(s) ; i++)
{
      if (s[i] >= 'A' && s[i] <= 'Z')
            s[i] = s[i] + ('a' - 'A');

      if (s[i] >= 'a' && s[i] <= 'z')
            s[i] = s[i] - ('a' - 'A');

      if (strchr("aeiouAEIOU", s[i]))
            s[i] = 'x';

}
```

a) *xdmxTxRx* **b)** *xdmxTxRe* **c)** *xDMxTxRe* **d)** *xDMxTxRx*

70. Considerăm că funcția *toupper* transformă o literă mică în literă mare. Care va fi valoarea șirului *s* după apelul *F(s,0,0)*, dacă inițial *s="ADmitEreATm"*?

```
void F(char* s, int n)
{
      if (n < 0)
            return;

      F(s, n - 1);

      if (n % 4)
            s[n] = toupper(s[n]);
}
```

a) ADMItEREATM
b) ADmItEReATM
c) ADMItEReATm
d) Niciuna dintre variantele de mai sus

71. Se consideră programul de mai jos. Menționați ce se va afișa în urma execuției acestuia.

```
void main(){
      char sir[40]="Este un exercitiu de programare";
      int n = strlen(sir);
      int i, j = 0;
      int m = 0, mi = 0, mj = 0;

      for (i = 0; i < n; i++) {
            if (sir[i] == ' ') {
                  if (i - j > m) {
                        m = i - j;
                        mi = i;
                        mj = j;
                  }
                  j = i + 1;
            }
      }
      char rez[30];
      strncpy(rez, sir + mj, mi);
      rez[mi - mj] = '\0';
      printf("%s", rez);
}
```

a) *programare* **b)** *exercitiu* **c)** *Acesta* **d)** *este*

72. Ce se va afișa în urma execuției programului definit mai jos?

```
void modify(char s[], int i, int l) {
      if (i >= l)
            return;

      if (s[i] % 2 == 0)
            s[i] = '*';
      else
            s[i] = '#';

      modify(s, i + 1, l);
}
void main() {
      char v[] = "12345678";
      modify(v, 0, sizeof(v) - 1);
      printf("%s", v);
}
```

a) *#*#*#*#**
b) *#**#**#***
c) **#*#*#*#**
d) **#*#*#*#*

73. Precizați valoarea variabilei *s* la finalul executării următoarei secvențe de program:

```
char s[100] = "aerisirea", aux[100];
for (int i = 0; i < strlen(s)/2; i++)
{
      if (strchr("aeiou", s[i])&&s[i]==s[strlen(s)-i-1])
      {
            strcpy(aux, s + i);
            strcpy(s + i + 1, aux);
            s[i + 1] = 'x';
            strcpy(aux, s + strlen(s) - i - 1);
            strcpy(s + strlen(s) - i, aux);
            s[strlen(s) - i - 1] = 'x';

            i++;
      }
}
```

a) xaxerxisxirxrxa
b) axerisireax
c) axexrisirexax
d) axexrixsixrexax

ALGORITMICĂ ȘI ANALIZA COMPLEXITĂȚII

74. Alegeți varianta care conține algoritmi în ordinea descrescătoare a complexităților considerând că sunt implementați eficient:

a) sortarea vectorilor, înmulțirea matricelor, căutarea binară, interschimbarea a două variabile

b) interschimbarea a două variabile, căutarea binară, înmulțirea matricelor, sortarea vectorilor

c) înmulțirea matricelor, sortarea vectorilor, căutarea binară, interschimbarea a două variabile

d) sortarea vectorilor, interschimbarea a două variabile, înmulțirea matricelor, căutarea binară

e) înmulțirea matricelor, căutarea binară, sortarea vectorilor, interschimbarea a două variabile

75. Precizați ce complexitate are algoritmul implementat de funcția *ex* definit mai jos, ținând cont că *n* este dimensiunea vectorului *v*, valorile din *v* sunt din intervalul $[0, 999]$ iar este *k* este numărul de valori distincte pe care le pot lua aceste valori.

```
void ex(int v[], int n)
{
      int i,j, numere[1000] = {0};

      for (i = 0; i < n; i++)
            numere[v[i]]++;

      for (i = 0; i < 1000; i++)
            for (j = 0; j < numere[i]; j++)
                  printf("%d ", i);
}
```

a) $O(n)$
b) $O(k)$
c) $O(n^2)$
d) $O(n * k)$

76. Se consideră subprogramul de mai jos. Care este complexitatea acestuia din punct de vedere al timpului de execuție?

```
void print(int n){
      int i;
      if (n == 0)
            return;
      for (i = 0; i < n; i++)
            printf("%d ", i);
      printf("\n");
      print(n - 1);
      printf("\n");
      for(i = 0;i<n;i++)
            printf("%d ", i);
}
```

a) $O(n)$
b) $O(n^3)$
c) $O(n^2)$
d) $O(n^2 \log(n))$

77. Dacă variabila *n* este de forma $n = 2^N$, precizați care este valoarea lui *sum* în funcție de N, după execuția secvenței de cod de mai jos:

```
int i, j, sum = 0;
for (i = 1; i <= n; i *= 2)
      for (j = 1; j < i; j *= 2)
            sum++;
```

a) $n = N$ **b)** $n = N + 1$ **c)** $n = \frac{2^N \cdot (2^N + 1)}{2}$ **d)** $n = \frac{N \cdot (N+1)}{2}$

78. Se consideră subprogramul de mai jos, care returnează numărul de drumuri unice dintr-o matrice cu *m* linii și *n* coloane, pornind de la colțul din stânga sus *(0,0)* până la colțul din dreapta jos $(m-1, n-1)$. Ce va returna apelul funcției calculează pentru $m = 3$ și $n = 3$?

```
int calculeaza(int m, int n) {
      if (m == 1 || n == 1)
            return 1;
      return calculeaza(m - 1, n) + calculeaza (m, n - 1);
}
```

a) 3 **b)** 6 **c)** 9 **d)** 20

79. Precizați care este șirul folosit ca parametru de intrare pentru apelul funcției *my_print* dacă la execuție funcția *my_print* afișează informația următoare:

a: ******
c: ***
e: ******
i: **
m: *
n: ***
p: *
r: ******
t: **
u: ***
z: *

```
void my_print(const char text[]) {
	int f[26] = { 0 }, i, j;

	for (i = 0; text[i] != '\0'; i++)
		f[(unsigned char)text[i] - 'a']++;

	for (i = 0; i < 26; i++)
	{
		if (f[i] > 0)
		{
			printf("%c: ", i + 'a');
			for (j = 0; j < f[i]; j++)
				printf("*");
			printf("\n");
		}
	}
}
```

a) "*computerele pot intelege zeci de numere*"
b) "*fiecare caracter este reprezentat in cod*"
c) "*reprezentarea numerica a unui caracter*"
d) "*caracterele neimprimabile nu au o reprezentare*"

80. Indicați ce valori afișează programul de mai jos:

```
int v[100], n=12, k;
void f(int i, int j){
	if (i * i > n)
		return;
	if (v[i] == 0)
	{
		if (j <= n / i)
		{
			v[i * j] = 1;
			f(i, j + 1);
			return;
		}
	}
	f(i + 1, 2);
}

void main(){
	v[0] = v[1] = 1;
	f(2, 2);
	for (k = 1; k < n; k++)
		if (v[k] == 0)
			printf("%d ", k);
}
```

a) 2 4 5 9 11
b) 2 3 5 7 11
c) 3 7 9 11 13
d) 2 3 7 9 11

81. Ce afișează programul de mai jos?

```
int f(int v[], int n) {
	int inv = 0;
	for (int i = 0; i < n; i++)
		for (int j = 0; j < n; j++)
			if ((i < j) && (v[i] > v[j]))
				inv++;
	return inv;
}
void main(){
	int v[] = { 2, 3, 8, 6, 1 };
	printf("%d", f(v, sizeof(v) / sizeof(v[0])));
}
```

a) 4 **b)** 5 **c)** 6 **d)** 7

82. Se consideră subprogramul de mai jos, unde *red, blue, wanted_red* și *tries* sunt numere întregi. Să se determine rezultatul returnat de execuția apelului *bile(3,3,2,3)*?

```
float bile(int red, int blue, int wanted_red, int tries){
      float tot = red + blue;
      if (wanted_red > red || tries < wanted_red)
            return 0;
      if (wanted_red == 0 && tries != 0)
            return blue / (float)(tot);
      if (wanted_red == 0 && tries == 0)
            return 1.0f;

      float p1 = red/tot*bile(red-1,blue,wanted_red-1,tries-1);
      float p2=blue/tot*bile(red,blue-1,wanted_red,tries-1);

      return p1 + p2;
}
```

a) 0.33 **b)** 0.45 **c)** 0.50 **d)** 0.76

83. Fie tabloul unidimensional $(12, 16, 19, 21, 24, 29, 30, 35, 80)$. Care este ordinea elementelor verificate pentru căutarea elementului $x = 85$ folosind metoda căutării binare?

a) 24, 30, 35, 80
b) 24, 30, 80
c) 24, 30, 35
d) 24, 35, 80

84. Precizați ce reprezintă valoarea întoarsă de funcția *func* definită mai jos, când este apelată pentru un vector *v* de *n* elemente întregi.

```
int func (int v[], int n){
      int i, j, my_values = 0, check;
      for (i = 0; i < n; i += 2)
      {
            check = 0;
            for (j = 0; j < n; j++)
            {
                  if (!(j % 2) && (i != j))
                        if (v[i] == v[j])
                              check = 1;
            }
            if (!check)
                  my_values++;
      }
      return my_values;
}
```

a) Numărul elementelor distincte din vectorul *v* având valori pare
b) Numărul elementelor distincte din vectorul *v*, aflate pe poziții pare
c) Numărul valorilor pare din vectorul *v*
d) Numărul elementelor distincte din vectorul *v*, aflate pe poziții impare

85. Considerăm secvența de numere întregi $A = \{1, 2, 1, 3, 4, 5, 9, 8\}$. Considerând funcția *g* de mai jos, ce va afișa apelul *g(A,8)*?

```
void g(int v[], int n){
	int lm = 1, lc = 1, m = 0, i;
	for (i = 1; i < n; i++)
	{
		if (v[i] > v[i - 1])
			lc++;
		else
		{
			if (lc > lm)
			{
				lm = lc;
				m = i - lc;
			}
			lc = 1;
		}
	}
	if (lc > lm)
	{
		lm = lc;
		m = n - lc;
	}
	for (i = m; i < m + lm; i++)
		printf("%d ", v[i]);
}
```

a) 1 3 4 5 9 **b)** 3 4 5 9 **c)** 1 2 1 3 **d)** 1 2

86. Considerăm două secvențe de numere întregi A = {1, 2, 3, 4, 5} și *B = {9, 8, 7, 5, 5}*. Utilizând subprogramul de mai jos, ce va afișa *f(A, B, 5)*?

```
void f(int A[], int B[], int n){
      int count = 0, i, j, t, k;
      for (i = 0; i < n; i++)
            for (j = 0; j < n; j++)
                  if (A[i] == B[j])
                  {
                        t = 0;
                        for (k = 0; k < i; k++)
                        {
                              if (A[k] == A[i])
                              {
                                    t = 1;
                                    break;
                              }
                        }
                        if (!t)
                        {
                              printf("%d ", A[i]);
                              count++;
                        }
                        break;
                  }
}
```

a) 1 2 3 4 5 **b)** 5 **c)** 1 2 3 **d)** 4 5

87. Indicați complexitatea secvenței de program de mai jos, considerând că *i, j, k, n* și *sum* sunt numere naturale întregi.

```
sum = 0;
for (i = 0; i < n; i++) {
      if (i == 1) {
            for (j = 0; j < n; j++)
                  for (k = 0; k < n; k++){
                        if (k % 2)
                              sum += k;
                        else
                              sum += i;
                  }
      }
}
```

a) $O(n)$ **b)** $O(nlogn)$ **c)** $O(n^2)$ **d)** $O(n^3)$

88. Decideți care dintre următoarele funcții este cea mai lentă, atunci când *n* devine foarte mare.

a)
```
void f(int n) {
      int i = 1;
      while (i < n) {
            printf("Work...\n");
            i *= 2;
      }
}
```

b)
```
void f(int n) {
    for (int i = 0; i < n; i++)
        printf("Work...\n");
}
```

c)
```
void f(int n) {
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            printf("Work...\n");
}
```

d)
```
void f(int n) {
    for (int i = 1; i < n; i++)
        for (int j = 0; j < n; j += i)
            printf("Work...\n");
}
```

89. Care este complexitatea subprogramului de mai jos?

```
int multiplication(int a, int e, int p)
{
      int result;
      if (!e)
            return 1;

      result = multiplication(a, e / 2, p);

      result = (result * result) % p;

      if (e % 2 == 1)
            result = (result * a) % p;

      return result;
}
```

a) O(a) **b)** O(log_2a) **c)** O(log_2e) **d)** O(a*log_2e)

90. Care este complexitatea următorului algoritm?

```
void f(int v[], int n) {
      for (int i = 0; i < n; i++)
            v[i] = i;
      for (int i = 0; i < n; i++)
            v[i] += 7;
      for (int i = 0; i < n; i++) {
            int x = 1;
            while (x < n)
                  x *= 4;
      }
}
```

a) $O(n)$
b) $O(nlog(n))$
c) $O(n^2)$
d) $O(2n + \log_4(n))$

91. Se consideră subprogramul de mai jos, care calculează cel mai mare divizor comun a două numere întregi, *a* și *b*. Considerăm *n=max(a,b).* Care este complexitatea temporală a acestui algoritm?

```
int cmmdc(int a, int b) {
    while (b != 0) {
        int r = a % b;
        a = b;
        b = r;
    }
    return a;
}
```

a) $O(n)$ **b)** $O(log\, n)$ **c)** $O(n\, log\, n)$ **d)** $O\left(\sqrt{n}\right)$

92. Precizați ce complexitate temporală are algoritmul implementat de subprogramul *operatii*, definit mai jos.

```
void operatii(int N){
      int i, numar_operatii = 0, auxiliar = N;
      while (auxiliar){
            for (i = 1; i * i <= N; i++)
                  numar_operatii++;
            auxiliar /= 3;
      }
}
```

a) $O(\log(N))$
b) $O(N * \log(N))$
c) $O(N^2 * \log(N))$
d) $O(\sqrt{N} * log_2 N)$

93. Precizați ce complexitate are algoritmul implementat de subprogramul *func21*, definit mai jos.

```
int func21(int n) {
      int x, s = 0, count = 0, i, j, k;

      for (i = s; i > s; i /= 2) {
            scanf("%d", &x);
            s += x;
      }
      for (i = n; i > 1; i /= 2)
            for (j = 0; j < n; j++)
                  for (k = 0; k < 1000; k++)
                        count++;
      return s + count;
}
```

a) $0(n^2)$ **b)** $0(nlog_2(n))$ **c)** $0(n)$ **d)** $0(\log_2(n))$

94. Se consideră o mulțime de numere naturale cu *n* elemente. Care este complexitatea temporală a unei funcții care returnează numărul de submulțimi ale mulțimii inițiale cu proprietatea că suma elementelor impare ale fiecărei submulțimi este divizibilă cu *n*, iar suma elementelor pare ale aceleiași submulțimi este divizibilă cu *(n+1)*.

a) $O(n)$
b) $O(2^n)$
c) $O(n * log(n))$
d) $O(log(n))$

95. Subprogramul de mai jos generează toate numerele prime mai mici decât *n*. Indicați complexitatea temporală a acestui subprogram, dacă este apelat pentru un vector *v* de dimensiune *n*, iar *v* conține doar valori naturale nule.

```
void genereaza(int v[], int n) {
      int p, i;
      v[0] = v[1] = 1;
      for (p = 2; p * p <= n; p++)
            if (v[p] == 0)
                  for (i = p * p; i < n; i += p)
                        v[i] = 1;

      for (i = 2; i < n; i++)
            if (v[i] == 0) printf("%d ", i);
}
```

a) $O(n)$
b) $O(2^n)$
c) $O(n * log(n))$
d) $O(n * log(log(n)))$

FUNCȚII RECURSIVE

96. Se consideră funcția *func*. Pentru ce pereche de valori ale parametrilor *a* și *b*, apelul funcției *func* va returna 25?

```
int func(int a, int b){
      if (a == 1 || b==1)
            return 1;
      if (a > b)
            return 1 + func(a / 2, b);
      if (a <= b)
            return 1 + func(a, b / 2);
}
```

a) a=2 b=4096
b) a=8192 b=4096
c) a=4096 b=2
d) a=2 b=25

97. Ce valoare este întoarsă de apelul funcției *f(7, 3)*?

```
int f(int n, int k) {
      if (k == 0 || k == n)
            return 1;

      return f(n - 1, k - 1) + f(n - 1, k);
}
```

a) 20 **b)** 21 **c)** 35 **d)** 56

98. Se consideră programul de mai jos. Ce afișează acesta pe ecran?

```
int funct(int n){
      if (n == 0)
            return 0;
      if (n % 2 == 0)
            return n * n + funct(n - 1);
      return (n - 1) * (n - 1) + funct(n - 1);
}
void main(){
      printf("%d", funct(101));
}
```

a) 343400 **b)** 333400 **c)** 343300 **d)** 858500

99. Se consideră funcția recursivă de mai jos, cu doi parametri de tip întreg. Care va fi a 5-a pereche afișată la apelul funcției *alg(15, 35)*?

```
int alg(int a, int b){
      int ca = a, cb = b;
      if (a == 1 || b > 100)
            return 0;

      while (ca != cb)
      {
            if (ca > cb)
                  ca = ca - cb;
            else
                  cb = cb - ca;
      }

      if (ca == 1)
            printf("(%d %d)\n", a, b);
      alg(--a, ++b);
}
```

a) (9 41) **b)** (3 47) **c)** (13 37) **d)** (7 43)

100. Subprogramul următor realizează sortarea unui vector de numere întregi. Cu ce expresie trebuie înlocuită zona marcată cu . . . . . , pentru ca subprogramul să îndeplinească în mod corect sarcina?

```
void fun(int v[], int n){
      int temp,i;
      if (n == 1)
            return;
      for (i = 0; i < n - 1; i++)
            if (v[i] > v[i + 1]) {
                  temp = v[i];
                  v[i] = v[i + 1];
                  v[i + 1] = temp;
      }
      . . . . . .;
}
```

a) *fun(v,n)-1* **b)** *fun(v,i)* **c)** *fun(v,i+1)* **d)** *fun(v,i-1)*

101. Ce valoare este afișată la execuția funcției *f*, dacă este apelată cu parametrii *n = 777*, *b = 5*?

```
void f(int n, int b)
{
	int r;
	if (n == 0)
		return;

	f(n / b, b);
	r = n % b;

	if (r < 10)
		printf("%d", r);
	else
		printf("%c", r - 10 + 'A');
}
```

a) 302 **b)** 1001210 **c)** 11102 **d)** 30021

102. Se dă vectorul *v[] = { 4, 7, 10, 5, 12, 3 }* și *n = 6*. Care este valoarea întoarsă de subprogramul *h(v, n)* definit mai jos?

```
int h(int v[], int n)
{
	if (n == 0)
		return 0;

	int x = v[n - 1];
	bool p = true;

	for (int k = 2; k * k <= x; k++)
	{
		if (x % k == 0)
		{
			p = false;
			break;
		}
	}
	int r = h(v, n - 1);

	if (p && x > 1)
		return r * 10 + x;
	else
		return r;
}
```

a) 357 **b)** 753 **c)** 73 **d)** 35

103. Se consideră funcția recursivă *f* definită mai jos. Câte caractere * și câte caractere # sunt afișate în cadrul apelului *f(100)*?

```
void f(int n){
      printf("*");
      if (n != 0)
      {
            printf("* ");
            if (n % 2 == 0){
                  printf("*");
                  f(n - 1);
            }
            else{
                  f(n - 1);
                  printf("#");
            }
      }
      else
            printf("*");
      printf("*  ");
}
```

a) 453 * 50 #
b) 450 * 50 #
c) 452 * 50 #
d) 403 * 50 #

104. Se consideră subprogramul de mai jos. Menționați care este valoarea returnată de funcția *g*.

```
int f(int n) {
      if (n == 0)
            return 0;
      if (n == 1)
            return 1;
      return (f(n - 1) + f(n - 2));
}
int g() {
      int i,j;
      for (i = 2; i < 30; i++) {
            int res = f(i);
            for (j = 1; j < res; j++)
                  if (j * j == res)
                        return res;
      }
      return 0;
}
```

a) 81 **b)** 610 **c)** 196 **d)** 144

105. Se consideră programul de mai jos. Ce se afișează pe ecran?

```
void fun(char* m, int sh, int i) {
      if (m[i] == '\0')
            return;
      m[i] = (m[i] - 'A' - sh + 26) % 26 + 'A';
      fun(m, sh, i + 1);
}
int main() {
      char v[] = "RNQNYFWDYJHMNSHFQFHFIJRD";
      fun(v, 5, 0);
      printf("%s\n", v);
      return 0;
}
```

a) MILITARYTECHNICALACADEMY
b) ACADEMIA TEHNICA MILITARA
c) ACADEMIATEHNICAMILITARA
d) TECHNICALMILITARYACADEMY

106. Fie funcția *fun* de mai jos. Ce valoare returnează apelul *fun(11)*?

```
int fun(int n) {
      if (n == 0)
            return 0;
      else
            if (n == 1)
                        return 1;
                  else
                        return 2 * fun(n - 1) + fun(n - 2);
}
```

a) 5741 **b)** 2378 **c)** 5740 **d)** 5471

107. Ce valoare este afișată în urma execuției programului de mai jos?

```
int getval(int x) {
      if (x == 0)
            return 3;
      else
            return getval(x - 1) + x;
}
int V[3000];
void main(){
      int  i;
      for (i = 0; i < 3000; i++)
            V[i] = getval(i);
      printf("%d", V[2025]);
}
```

a) 6075 **b)** 6078 **c)** 2049303 **d)** 2051328

108. Se consideră subprogramul de mai jos, unde *n* reprezintă un întreg. Menționați care este rezultatul apelului funcției *f(28)*.

```
int f(int n){
      if (n <= 9)
            return 0;
      if (n % 10 > 5)
            if ((n / 10) % 2 == 0)
                  return f(n - 1) + n;
      return f(n - 1);
}
```

a) 151 **b)** 106 **c)** 0 **d)** 81

109. Dacă pentru secvența de cod de mai jos, apelul subprogramului *f* returnează valoarea *17*, ce valori inițiale au variabilele *n*, *a* și *b* în această ordine?

```
int f(int n, int a, int b){
      if (n == 0)
            return 1;
      else if (n == 1)
            return a + b;
      else
            return a * f(n-1, a, b) + b * f(n-2, a, b);
}
```

a) 4 1 1 **b)** 3 4 1 **c)** 3 2 1 **d)** 3 1 2

110. Considerând variabila globală *a* un vector de numere întregi astfel încât *a[i] = i*, primul index fiind 1 și variabila întreagă n, numărul de elemente al vectorului, ce valoare va avea apelul funcției *f(1)*?

```
int f(int i){
      if (i > n)
            return 0;
      int s = 0;
      for (int j = 1; j <= a[i]; j++)
            s += j;

      return s + f(i + 1);
}
```

a) $\frac{1}{6}n(n+1)(n+2)$

b) $\frac{1}{6}n(n+1)(4n+1)$

c) $n(n+1)(n+2)$

d) $\frac{1}{2}n(n+1)(n+2)$

111. Precizați ce valoare returnează apelul funcției *f(7125142,1)*, unde funcția *f* este definită mai jos:

```
int f(int n, int c){
      if (n < 10)
            return n;
      if ((n % 10) == c + 1 && ((n / 10) % 10) == c)
            return f(n / 100, c);
      else
            return f(n / 10, c) * 10 + (n % 10);
}
```

a) 71142 **b)** 75142 **c)** 71512 **d)** 725142

112. Precizați ce valoarea întoarce apelul *f(mat, 3, 0, 0)* pentru funcția *f* definită mai jos:

```
int mat[3][3] = { {1,2,3},{4,5,6},{7,8,9} };

int f(int mat[][3], int n, int i, int j){
      int sum = 0;
      if (i >= n) return -1;
      if (i + j <= n - 1)
            sum += mat[i][j];

      if (!(j == n - 1))
            return sum + f(mat, n, i, j + 1);
      else
            return sum + f(mat, n, i + 1, 0);
}
```

a) 7 **b)** 21 **c)** 6 **d)** 22

113. Precizați ce valoare returnează apelul *f(15247,0)*, pentru funcția *f* definită mai jos:

```
int f(int nr, int value){
      if (nr == 0)
            return value;

      int a = value * 10 + nr % 10;
      return f(nr / 10, a + nr%10);
}
```

a) 74251 **b)** 78451 **c)** 144501 **d)** 148502

114. Precizați ce valoare întoarce apelul funcției f(v, n, 1) pentru v și n definite mai jos:

```
int v[] = { 48, 61, 42, 86, 35, 74, 68, 12, 13, 25 };
int n = sizeof(v) / sizeof(v[0]);

int f(int v[], int n, int index) {
      if (index >= n)
            return 0;

      if (index % 3 == 0)
            return f(v, n, index + 1) - v[index];
      else
            return f(v, n, index + 1) + v[index];
}
```

a) 100 **b)** 78451 **c)** 144501 **d)** 148502

115. Precizați care este suma elementelor matricei *matrice* după execuția apelului *f(matrice, 3, 3, 5, 0, 0)*, aplicat pentru variabila *matrice* definită mai jos:

```
int matrice[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
void f(int matrice[][3],int m,int n,int p,int i,int j){
      if (i >= m)
            return;
      if (j >= n)
      {
            f(matrice, m, n, p, i + 1, 0);
            return;
      }
      if (matrice[i][j] < p)
            matrice[i][j] = -1;
      f(matrice, m, n, p, i, j + 2);
}
```

a) 39 **b)** 43 **c)** 45 **d)** 34

116. Se consideră subprogramul de mai jos. Menționați pentru ce valoare a lui *x* apelul funcției *f(x,4)* va returna valoarea *6*.

```
int f(int x, int y){
      if ((!x) || (!y))
            return 1;
      if (y % 2 == 0)
            return f(x, y - 1) + f(x - 1, y);
      if (x % 2)
            return f(x - 1, y) + f(x, y - 1);
      return 1;
}
```

a) 0 **b)** 1 **c)** 2 **d)** 3

117. Indicați ce va afișa programul de mai jos.

```
int program(int x, int y, int z) {
      int n=0;
      if (x*x + y*y == z)
            n++;
      if (z>500)
            return 0;

      return n + program(x+y, y+1, z+z) + x+y;
}
void main(){
      printf("%d", program(13, 0, 50));
}
```

a) 63 **b)** 113 **c)** 62 **d)** 114

118. Se consideră subprogramul de mai jos. Ce se va afișa în urma apelului f(8)?

```
int f(int n){
      if (!n) return 0;
      if (n % 2 == 0)
      {
            printf("%d", n);
            return f(n + 1) + f(n - 1);
      }
      else
      {
            printf("%d", n);
      return f(n / 2);
      }
}
```

a) 8943512131731
b) 8943152131731
c) 8974353121311
d) 8945231131731

119. Se consideră subprogramul de mai jos:

```
void f(int a, int b){
    if (b == 0) return;
    f(b, a - b);
    printf("%d ", a - b);
}
```

Considerând apelul *f(610, 377)*, câte dintre afirmațiile următoare sunt adevărate?

I. Codul va afișa un șir de 14 numere distincte.
II. Codul va afișa șirul lui Fibonacci în ordine descrescătoare.
III. Ultimul element afișat este 0.
IV. Primul element afișat este 377.

a) 0 **b)** 1 **c)** 2 **d)** 3

120. Indicați ce afișează programul de mai jos.

```
int f(int a, int b, int c) {//b si c sunt cifre
      if (a == 0)
            return 0;
      int rezultat = f(a / 10, b, c);
      if (a % 10 == c)
            rezultat = rezultat * 100 + b * 10 + c;
      return rezultat;
}

void main() {
      int a = 51212213;
      int nr = f(a, 4, 1) + f(a, 5, 3);
      printf("%d", nr);
}
```

a) 0 **b)** 414141 **c)** 414194 **d)** 51212217

121. Se consideră programul de mai jos, unde *x* reprezintă un număr întreg. Ce valori pot fi introduse pentru *x* astfel încât la final, programul să afișeze, în această ordine, valorile: *0, 1, 1*.

```
int f(int n) {
      if (n == 0)
            return 1;
      int c1 = n % 10;
      int c2 = n / 10 % 10;

      return f(n / 10) && (c1 > c2);
}

void main() {
      int x;
      for (int i = 0; i < 3; i++) {
            scanf("%d", &x);
            printf("%d\n", f(x));
      }
}
```

a) 4321, 256, 7 **b)** 89, 143, 765 **c)** 632, 532, 981 **d)** 245, 689, 23

122. Ce valoare va fi returnată la apelul funcției *f(112446939)*?

```
int g(int a, int b){
      while (a != b)
      {
            if (a > b) a -= b;
            else b -= a;
      }
      return a;
}
int f(int x){
      int d, u;
      if (x / 10 == 0)
            return 0;
      d = (x / 10) % 10;
      u = x % 10;
      return f(x / 100) + g(d, u);
}
```

a) 8 **b)** 9 **c)** 11 **d)** 10

123. Se consideră subprogramul de mai jos, unde *n* este un număr întreg. Indicați ce va returna apelul funcției *func(*1299540087*)*.

```
int func(int n){
      int c, i, j, ok;
      int s = 0;
      if (n == 0)
            return 1;
      c = n % 10;
      for (i = 2; i <= c / 2; i++)
            if (c % i == 0)
                  s += i;
      ok = 0;
      for (j = 1; j * j <= c; j++)
      {
            if (j * j == c){
                  ok = 1;
                  break;
            }
      }
      if (ok)
            s += 1;
      if (s == 0)
            return func(n / 10);
      else
            return s * func(n / 10);
}
```

a) 1 **b)** 288 **c)** 0 **d)** 576

124. Se consideră programul de mai jos, precizați care set de valori poate fi introdus pentru n astfel încât de fiecare dată să se afișeze "DA".

```
int func(int n, int rez) {
      if (n == 0)
            return rez;
      return func(n / 10, rez * 10 + n % 10);
}
void main() {
      int n = 0, result = 0;
      scanf("%d", &n);
      result = func(n, 0);
      if (result == n)
            printf("DA");
}
```

a) 12321, 4567654, 344443
b) 19931, 12321, 946244
c) 89898, 475395, 125521
d) 94783, 8462578, 593758

125. Considerând funcția *F* definită mai jos, care va fi valoarea șirului de caractere *s* inițializat cu *"AdmitereATM"* după apelul *F(s, 0, 0)?*

```
void F(char* s, int i, int j) {
      if (s[i] == '\0') {
            s[j] = '\0';
            return;
      }
      if (strchr("aeiou", s[i]))
            F(s, i + 1, j);
      else {
            s[j] = s[i];
            F(s, i + 1, j + 1);
      }
}
```

a) *dmtrTM* **b)** *AieeA* **c)** *AdmtrATM* **d)** *dmitereTM*

126. Indicați ce valoare întoarce funcția de mai jos considerând că *n* și *x* sunt două numere naturale nenule.

```
int f (int n, int x)
{
      if (n == 0) return 1;
      else if (x > n) return 0;
      else return f(n – x,  x + 3);
}
```

a) 1 dacă există un număr natural k a.î. $n = \frac{k\,(2x+3k+1)}{2}$ sau 0 în caz contrar

b) 1 dacă există un număr natural k a.î. $n = \frac{k\,(2x+3k+1)}{2}$ sau 0 în caz contrar

c) 1 dacă n este divizibil cu 3 sau 0 în caz contrar

d) 1 dacă x este divizibil cu 3 sau 0 în caz contrar

127. Se consideră funcția *set* definită mai jos. Precizați ce valoare are al *30*-lea element din vectorul v populat cu *100* elemente prin apelul *set(v,100,0,3).*

```
void set(int v[], int n, int i, int last){
      int j;
      if (i == n) return;
      if (i == 0)
            v[i] = last;
      else{
            v[i] = 0;
            for (j = 1; j <= i + 1; j++)
                  v[i] += last + 2 * j;
            last = last + 2 * (j - 1);
      }
      set(v, n, i + 1, last);
}
```

a) 900 **b)** 960 **c)** 27060 **d)** 29853

128. Pentru implementarea unor operații cu elemente, perechi de numere întregi, se definesc doi vectori *v_primul* și *v_ultimul*, după cum urmează:

```
int v_primul[3000], v_ultimul[3000];
```

Precizați ce valori au câmpurile elementului al *2018*-lea aflat în fiecare dintre cei doi vectori populați prin apelul `init_v(v_primul, v_ultimul, 3000, 0)`, unde *init_v* este funcția definită în secvența de cod următoare:

```
void init_v(int v_primul[], int v_ultimul[],int n,int i){
      if (i == n) return;
      if (i == 0) {
            v_primul[i] = 1;
            v_ultimul[i] = 3;
      }
      else {
            v_primul[i] = v_primul[i - 1] + 3;
            v_ultimul[i] = v_ultimul[i - 1] + 4;
      }
      init_v(v_primul, v_ultimul, n, i + 1);
}
```

a) 6052; 8071

b) 6049; 8067

c) 6054;8072

d) 6051; 8068

129. Precizați ce valoare întoarce apelul *f (v, 0, 199)* aplicat pe un vector *v* de *200* de elemente întregi inițializat cu apelul *init(v, 200)*, unde:

```
void init(int v[ ], int n){
      int i = 0;
      do {
            i++;
            v[i-1] = i;
      }  while (i < n);
}
int f(int v[ ], int i, int j){
      if (i == j)
            return v[i] * v[i];
      return f(v, i, (i+j) / 2) + f(v, (i+j) / 2 + 1, j);
}
```

a) 19900 **b)** 20100 **c)** 2686700 **d)** 2727101

130. Se consideră funcția *F* de mai jos. Care va fi valoarea șirului de caractere *s* inițializat cu *"AdmitereATM"* după apelul *F(s, 3, 2)*?

```
void F(char s[], int k, int c) {
      int n = strlen(s);
      if (c == 0)
            return;
      char tmp[50];
      int r = k % n;
      for (int i = 0; i < n - r; i++)
            tmp[i] = s[i + r];
      for (int i = 0; i < r; i++)
            tmp[n - r + i] = s[i];
      tmp[n] = '\0';
      for (int i = 0; i < n; i++)
            s[i] = tmp[i];
      F(s, k, c - 1);
}
```

a) ereATMAdmit
b) itereATMAdm
c) reATMAdmite
d) Niciuna dintre variantele de mai sus

BACKTRACKING ȘI PROBLEME DE GENERARE

131. Un program generează secvențe de *0* și *1*, folosind metoda *backtracking*, după o serie de reguli. O secvență este validă dacă respectă cumulativ următoarele cerințe:

1. este formată doar din cifrele *0* și *1* și are lungimea de zece cifre;
2. nu conține două cifre consecutive de *1*;
3. conține numai grupuri de cifre de *0* de lungime impară.

Programul a generat primele patru secvențe valide în următoarea ordine: *[0000000001], [0000000101], [0000010001], [0000010101]*.

Alegeți afirmația corectă referitoare la secvențele generate de program.

a) numărul de secvențe valide ce conțin exact *5* cifre de *1* este *3*
b) a *7*-a secvență validă este *[0100010101]*
c) exista secvențe valide ce conțin exact *6* cifre de *1*
d) numărul de secvențe valide ce conțin exact *5* cifre de *0* este *2*

132. Se consideră următorul subprogram unde *s* este un șir de caractere, iar *st* și *dr* sunt două valori întregi. Indicați ce va returna apelul funcției *p* pentru valorile *s = "abc", st = 0* și *dr = 2*.

```
void p(char s[], int st, int dr) {
      if (st == dr) {
            printf("%s, ", s);
            return;
      }
      for (int i = st; i <= dr; i++) {
            char temp = s[st];
            s[st] = s[i];
            s[i] = temp;

            p(s, st + 1, dr);

            temp = s[st];
            s[st] = s[i];
            s[i] = temp;
      }
}
```

a) abc, bac, bca, cba, cab,
b) abc, acb, bac, bca, cab, cba,
c) abc, acb, cab,
d) abc, bac,

133. Se consideră toate numerele de trei cifre care sunt generate în ordine, folosind cifrele din mulțimea $\{0, 1, 2, 3, 4, 5\}$ cu mențiunea că două cifre alăturate nu pot avea aceeași paritate. Primele patru numere generate sunt: *101*, *103*, *105* și *121*. Care sunt al 7-lea și penultimul număr din secvența generată?

a) 141, 543 **b)** 125, 541 **c)** 143, 543 **d)** 301, 525

134. Folosind metoda *backtracking*, s-au generat numere de maximum *4* cifre, formate din cifrele 1, 2, 3, 4 și 5 care au produsul cifrelor în intervalul [*7*, *11*]. Fiecare cifră poate să apară maximum o dată în cadrul unui număr generat. Primele *4* numere generate sunt: 124, 125, 142 și 152. Care vor fi următoarele *4* numere generate?

a) 214, 251, 24, 412
b) 214, 215, 24, 412
c) 214, 215, 24, 241
d) 214, 251, 241, 24

135. Vectorul *A* este inițializat cu cifre folosind funcția *set_vector*, definită mai jos. Câte cifre de *8* sunt în vectorul *A* dacă acesta este inițializat cu exact *55* cifre.

```
void set_vector(int A[], int n) {
     int i, j, cont = 0;
     for (i = 1; i <= n  && cont < n; i++)
     {
          j = 1;
          while (j <= i) {
               A[cont++] = 8;
               j++;
               if (cont == n) break;
          }
          if (cont == n) continue;
          A[cont++] = 9;
     }
}
```

a) 45 **b)** 46 **c)** 47 **d)** 55

GRAFURI NEORIENTATE

136. Se consideră un graf neorientat, reprezentat prin matricea de adiacență de mai jos. Selectați afirmația adevărată.

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

a) Adăugând muchia (3,7), graful devine Hamiltonian
b) Graful admite 256 grafuri parțiale
c) Graful are 9 muchii
d) Graful are o singură componentă conexă

137. Dacă *A* este matricea de adiacență a unui graf, atunci elementul *(i,j)* al matricei A^2 reprezintă:

a) Numărul de muchii care ies din nodul *i*
b) Numărul de drumuri de lungime *2* de la nodul *i* la nodul *j*
c) Numărul de drumuri de lungime *1* de la nodul *i* la nodul *j*
d) *1* dacă există un muchie directă $[i, j]$, altfel *0*

138. Se consideră un graf neorientat cu noduri numerotate de la *1* la *10*, având asociat matricea de adiacență de mai jos. Câte componente conexe va avea graful, dacă din configurația curentă eliminăm toate muchiile dintre oricare două perechi de noduri formate dintr-un nod par și un nod impar?

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

a) 5 **b)** 2 **c)** 1 **d)** 4

139. Care este numărul de grafuri neorientate cu mulțimea nodurilor *{1,2,3,4,5,6,7,8}*, în care nodul 5 este izolat, iar nodul 2 are gradul maxim posibil?

a) 2^{15} **b)** 2^{21} **c)** 4^{11} **d)** 2^{27}

140. Se consideră subprogramul de mai jos, care afișează toate perechile de noduri neadiacente ale unui graf neorientat. Care este modul corect de apelare pentru care subprogramul execută corect sarcina implementată, știind că *adj* este o matrice de adiacență a unui graf neorientat cu *8* noduri?

```
void fun(int adj[8][8], int n, int i, int j) {
      if (i == n)
            return;
      if (j == n) {
            fun(adj, n, i + 1, i + 1);
            return;
      }
      if (i < j && adj[i][j] == 0)
            printf("(%d, %d)\n", i, j);
      fun(adj, n, i, j + 1);
}
```

a) *fun(adj, 8, 1, -1)*
b) *fun(adj, 8, 0, 1)*
c) *fun(adj, 8, 8, 8)*
d) *fun(adj, 8, 1, 0)*

141. Se consideră graful *G* cu *6* noduri, toate având gradul *2*. Ce se poate afirma cu certitudine despre graful *G* ?

a) Graful este eulerian
b) Graful este hamiltonian
c) Graful are o singură componentă conexă
d) Graful conține un subgraf eulerian

142. Fie un graf *G* cu 6 noduri. Care dintre următoarele variante poate reprezenta gradele celor 6 noduri, știind că graful G este hamiltonian?

a) 2 1 4 1 4 4
b) 4 3 3 2 2 3
c) 2 4 2 2 3 3
d) 5 3 4 4 1 5

143. Se consideră funcția *isPrime* având prototipul *int isPrime(int n)*, care întoarce valoarea *1* dacă *n* este prim și valoarea *0* altfel. Se consideră și funcția *getdegree* având prototipul *int getdegree(int v)*, care întoarce pentru graful *G*, gradul nodului *v*. Considerăm *N* numărul de noduri al grafului *G*. Indicați pentru care dintre grafurile oferite prin matricea de adiacență codul de mai jos afișează mesajul *"Graful este special!"*.

```
int test(int N){
      int i;
      for (i = 0; i < N; i++)
            if (isPrime(getdegree(i))==1&&getdegree(i)>0)
                  return 0;
      printf("Graful este special!");
      return 1;
}
```

a) $\begin{pmatrix} 0&1&0&1 \\ 1&0&0&1 \\ 0&0&0&1 \\ 1&1&1&0 \end{pmatrix}$ **b)** $\begin{pmatrix} 0&1&1&1 \\ 1&0&0&1 \\ 1&0&0&0 \\ 1&1&0&0 \end{pmatrix}$ **c)** $\begin{pmatrix} 0&1&0&0&0 \\ 1&0&0&0&0 \\ 0&0&0&1&0 \\ 0&0&1&0&0 \\ 0&0&0&0&0 \end{pmatrix}$ **d)** $\begin{pmatrix} 0&1&1&1 \\ 1&0&0&1 \\ 1&0&0&1 \\ 1&1&1&0 \end{pmatrix}$

144. Fie un graf neorientat complet format din *n* noduri. Care este numărul de muchii din graf și care este numărul de grafuri parțiale care se pot obține din graful inițial dacă $n = 7$?

a) 21 muchii; 21 grafuri parțiale
b) 42 muchii; 2^{42} grafuri parțiale
c) 21 muchii; 2^{21} grafuri parțiale
d) 42 muchii; 42 de grafuri parțiale

145. Fie un arbore binar cu *N* noduri. Se consideră că un arbore binar este un arbore în care fiecare nod are *0, 1* sau *2* copii. Care este înălțimea arborelui, știind că acesta are exact un nod de tip frunză.

a) N-1 **b)** N/2 **c)** log(N) **d)** (N-1)/2

146. Se consideră un graf neorientat cu *8* noduri numerotate de la *1* la *8* definit prin lista de muchii [1,2], [1,3], [1,8], [2,6], [2,7], [3,4], [3,5]. Care dintre următoarele afirmații sunt adevărate?

A. Graful este aciclic
B. Graful conține un ciclu eulerian
C. Graful conține un ciclu hamiltonian
D. Graful este conex

a) A, D **b)** B,C **c)** A,C **d)** B

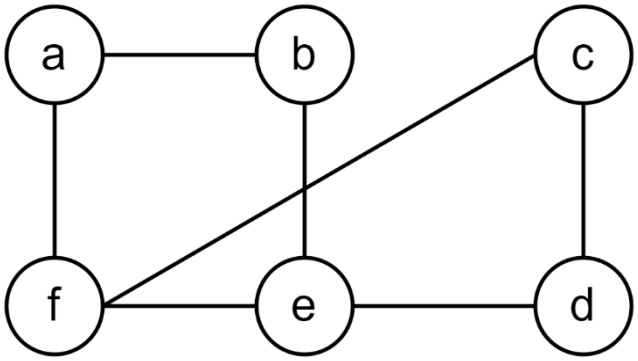
147. Se dă un graf neorientat definit de nodurile $V = \{ 1, 2, 3, ..., 2024 \}$ și setul de muchii $E = \{(x, x + 1010)\}$, astfel încât $x \in V$ și $x + 1010 \in V$. Care propoziție este adevărată?

a) Graful conține un ciclu
b) Graful are cel puțin 1000 de muchii
c) Graful este conex
d) Gradul cel mai mare al unui nod este 3

148. Fie $G = (V, A)$ un graf neorientat complet cu $n = 100$ vârfuri. Care dintre următoarele afirmații este *falsă*?

a) Graful G este hamiltonian
b) Graful G este eulerian
c) Graful are 4950 de muchii
d) Dacă se elimină 50 de muchii graful poate deveni eulerian

149. Considerând graful de mai jos, selectați afirmația corectă:



a) Suma gradelor este 14; graful este conex și conține un ciclu eulerian
b) Suma gradelor este 14; graful este conex și nu conține un ciclu eulerian
c) Suma gradelor este 12; graful este conex și conține un ciclu eulerian
d) Suma gradelor este 12; graful nu este conex și nu conține un ciclu eulerian

ARBORI

150. Un arbore cu 10 noduri, numerotate de la 1 la 10, are următorul vector de tați :[2,0,2,3,3,4,4,5,5,6]. Păstrând nodurile și muchiile, se dorește minimizarea înălțimii arborelui. Care dintre noduri ar fi cea mai potrivita alegere pentru a deveni noua rădăcină?

a) 1 **b)** 4 **c)** 2 **d)** 7

151. Se consideră un arbore cu proprietățile următoare:

- fiecare nod intern are 2^n copii.
- toate nivelurile arborelui sunt pline.
- h este înălțimea arborelui.

Care este numărul maxim de noduri pe care le poate avea arborele?

a) $\frac{(2^n)^{h+1}-1}{2^n-1}$

b) $(2^n)^{h+1}$

c) $2^n \cdot h$

d) $2^n(h+1)$

152. Precizați care este lungimea maximă a unui lanț elementar ce leagă două noduri într-un arbore cu 24 de noduri, dacă fiecare nod intern are gradul 3 (mai puțin frunzele, care pot avea și alt grad).

a) 7 **b)** 8 **c)** 9 **d)** 6

153. Se consideră arborele cu *6* noduri, notate de la *1* la *6*, reprezentat prin matricea de adiacență definită mai jos. Care este nodul ales ca rădăcină astfel încât înălțimea arborelui să fie minimă?

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

a) 4 **b)** 2 **c)** 1 **d)** 3

154. Se consideră un arbore cu 10 noduri, numerotate de la 1 la 10 având vectorul de tați următor: $(5, 3, 7, 7, 0, 5, 5, 3, 5, 4)$. Lungimea maximă a unui lanț elementar ce leagă două noduri ale acestuia este:

a) 4 **b)** 5 **c)** 2 **d)** 3

155. Se consideră subprogramul de mai jos, unde *tata* este un vector de valori întregi ce conține reprezentarea arborelui sub formă de vector de tați, iar *nr* numărul de noduri. Ce va întoarce subprogramul *drum* în cazul unui arbore, folosind ca parametri vector de tați *tata* și numărul de noduri *nr*.

```
void drum(int tata[], int nr){
      int i, maxim = 0, nod = -1;
      for (i = 0; i < nr; i++)   {
            int parinte = tata[i];
            int cale = 0;
            while (parinte != -1)
            {
                  cale++;
                  parinte = tata[parinte];
            }
            if (cale > maxim)
            {
                  maxim = cale;
                  nod = i;
            }
      }
      return maxim;
}
```

a) nodul cel mai îndepărtat de rădăcină
b) numărul de noduri deconectate de restul arborelui
c) numărul de frunze
d) înălțimea arborelui

PROBLEME DE NIVEL MEDIU-DIFICIL. SOLUȚII

1. Secvența de program va afișa unul dintre numerele aflate în vectorul *a*. Indexul elementului este aflat prin scăderea caracterelor *'j'* și *'i'*. Cum caracterele *i* și *j* sunt consecutive în tabela ASCII, expresia *'j'-'i'* este evaluată la valoarea 1.
Observație: diferența nu este calculată între variabilele *j* și *i*, ci între două constante, și anume caracterele *'j'* și *'i'*.

Răspunsul corect este **c)**.

2. Vectorul *reprezentare* este completat cu elementul de pe poziția din alfabet corespunzătoare restului împărțirii. La final, caracterele sunt afișate în ordine inversă. Funcția *f* afișează parametrul *numar* convertit în baza menționată prin parametrul *baza*.

Răspunsul corect este **b)**.

3. Identificăm contraexemple pentru fiecare dintre cele patru condiții:
Pentru $a = 11$ și $b = 4$, varianta *a* este falsă.
Pentru $a = b$, varianta *b* este falsă.
Varianta *c* este corectă.
Pentru $a = b$, varianta *d* este falsă.

Răspunsul corect este **c)**.

4. Doar varianta *b* verifică, pe rând, dacă două dintre variabile sunt egale și în același timp diferite de cea de-a treia.

Răspunsul corect este **b)**.

5. Expresia `oper >= 1 && oper <= 2` face ca funcția *operatie* să fie apelată numai dacă variabila *oper* are valorile *1* sau *2*. Cum *oper* are valoarea *3*, funcția nu va fi apelată și va fi asignată valoarea *-999*.

Răspunsul corect este **d)**.

6. Prima buclă *while* numără de câte ori apare fiecare cifră în *n* și stochează această statistică în vectorul *f*. A doua buclă *while*, determină cifra cu frecvența maximă, adică indexul la care se află cea mai mare valoare în *f*.

Răspunsul corect este **c)**.

7. În urma execuției celor două bucle for se obține ciurul lui Eratostene, unde $v[p] = \begin{cases} 0, & p\ este\ prim \\ 1, & p\ nu\ este\ prim \end{cases}$

Răspunsul corect este **a)**.

8. Subprogramul calculează pe rând numerele de două cifre consecutive din *x*, apoi verifică dacă numărul obținut este mai mare decât *50*. În cazul de față, numerele de două cifre sunt $\{51, 65, 46, 24, 92, 29\}$ dintre care *3* sunt mai mari decât *50*.

Răspunsul corect este **b)**.

9. Programul pornește de la *1* și incrementează aux cu *0.01*, apropiindu-se astfel, pas cu pas de rădăcina pătrată a lui *x*. Când diferența dintre *aux* și valoarea anterioară ($aux - 0.1$) începe să crească în loc să scadă, însemnă că am trecut de valoarea optimă și ne oprim, întorcând numărul pentru care diferența absolută dintre pătratul său și *x* este minimă.

Răspunsul corect este **c)**.

10. Din variabila *n* programul selectează ultima cifră, apoi ultimele două cifre, ultimele trei cifre, ș.a. La variabila *s* (care va fi afișată) se adaugă numerele formate dintr-un număr impar de cifre și se scad numerele formate dintr-un număr par de cifre.

$$s = 3 - 43 + 643 - 5643 + 45643 - 345643 = -305040$$

Răspunsul corect este **b)**.

11. Prima buclă for adună la z pe x de $\frac{y}{2}$ ori, adică z devine $\frac{x*y}{2}$.
A doua buclă for, adună la z pe y, de $x * y$ ori, adică adună $x * y^2$.
Astfel, *f* returnează $\frac{x*y}{2} + x * y^2$.

Răspunsul corect este **b)**.

12. Programul numără de câte ori apare valoarea din variabila *a* în variabila *b*, reprezentate în format zecimal. Programul iterează prin cifrele numărului *b*, de la dreapta la stânga, comparându-le cu cifrele numărului *a*.

Răspunsul corect este **a)**.

13. Funcția `pow(m, n)` ridică numărul *m* la puterea *n*. Astfel, fiecare element al matricei va fi puterea lui doi corespunzătoare sumei indecșilor de pe linii și coloane.
Chiar dacă pentru rezolvarea problemei nu este necesară construirea matricei, pentru o mai bună înțelegere ea poate fi reprezentată astfel:

$$\begin{pmatrix} 1 & 2 & 4 & 8 & 16 \\ 2 & 4 & 8 & 16 & 32 \\ 4 & 8 & 16 & 32 & 64 \\ 8 & 16 & 32 & 64 & 128 \\ 16 & 32 & 64 & 128 & 256 \\ 32 & 64 & 128 & 256 & 512 \end{pmatrix}$$

Valorile adunate pot fi calculate astfel:
$matrix[4]0] = 2^{(4+0)} = 16$
$matrix[4]0] = 2(^{1+3)} = 16$
$matrix[4]0] = 2^{(5+2)} = 64$

Variabila *sum* va avea valoarea *160*.

Răspunsul corect este **c)**.

14. Matricea pătratică *mat* este generată pe baza formulei $(i + j)^2 \% n$, unde *n* este parametrul de intrare iar *i* și *j* au valori în intervalul $[0, n)$.
Pentru $n = 4$, este afișată matricea de la varianta c.

Răspunsul corect este **c)**.

15. Programul verifică întâi dacă suma indicilor este pară `((i+2)%2==0)`, apoi exclude elementele de pe diagonala principală `(i !=j)` și apoi pe cele de pe diagonala secundară `(i+j != n+1)`.

Răspunsul corect este **a)**.

16. Funcția verifică paritatea elementelor consecutive în vectorul *v* până la poziția *n*. Condiția pentru a returna *1* este ca parametrul *n* să fie mai mic decât 2 sau primele *n* elemente din vectorul *v* să respecte un șablon de tipul par, impar, par, impar, etc. Primele 5 elemente din vectorul declarat la răspunsul b) respectă a doua condiție. Chiar dacă vectorul conține mai multe elemente, funcția va verifica doar *n*.

Răspunsul corect este **b)**.

17. Funcția calculează *cmmdc* între primele două elemente ale vectorului. Apoi, calculează cmmdc între rezultatul obținut și următorul element din vector ș.a.m.d. Rezultatul final este cmmdc al tuturor elementelor din vector.

Răspunsul corect este **b)**.

18. Programul extrage pe rând fiecare cifră a lui *N* și calculează o sumă. În calcularea sumei intră două tipuri de cifre ale lui *N*: cifrele pare de pe poziții impare și cifrele impare de pe poziții pare. Această sumă trebuie să fie 10.

Răspunsul corect este **d)**.

19. Programul inversează numărul și elimină cifrele pare. Operația de adăugare a cifrei (`r=r*10+x%10`) se realizează doar pentru cifrele impare.

Răspunsul corect este **d)**.

20. Programul elimină cifrele din variabila *b* care se regăsesc în variabila *a* și le dublează (adică apar de 2 ori) pe cele impare care nu se găsesc în *a*.

Răspunsul corect este **c)**.

21. Programul iterează printre cifrele numărului *n,* începând cu ultima cifră, din *2* în *2* cifre. Astfel, se formează oglinditul numărului *n* doar din cifre impare. La întâlnirea unei cifre pare, bucla *while* își încheie execuția.

Răspunsul corect este **a)**.

22. Programul contorizează numărul de cifre pare al elementelor vectorului dat. Ulterior, verifică dacă numărul astfel obținut este palindrom.

Răspunsul corect este **d)**.

23. Inițial, în variabila *my_value* se calculează suma cifrelor variabilei *a* și se obține 26. Apoi, din nou, în *my_value* se calculează suma cifrelor lui 26 și se obține 8. La final, *my_value* are valoarea 8 așadar, are o singură cifră.

Răspunsul corect este **d)**.

24. Funcția generează elementele din șirul lui Fibonacci strict mai mici decât *1547*. Fiecare element generat este verificat dacă este prim, adică dacă are maximum doi divizori. Funcția contorizează numărul de elemente Fibonacci strict mai mici decât *1547* și prime.

Răspunsul corect este **d)**.

25. Condiția impusă în instrucțiunea *if* face ca apelul menționat să afișeze perechi $(v[i], v[j])$ pentru care $i < j, v[i] < v[j]$, iar $v[i]$ este divizibil cu *2* și $v[j]$ este divizibil cu *3*.

Sunt afișate 3 perechi*:* $(2,3), (2,6)$ *și* $(4,6)$

Răspunsul corect este **d)**.

26. Variabila *i* ia pe rând valorile {1,2,4,8}.
Variabila *j* ia pe rând valorile {16,8,4,2,1}.
Pentru fiecare dintre aceste valori, numărul de iterații ale celui de-al treilea ciclu *for* (după k) este $[(n - j + 1)/2]$.
Numărul de iterații total este: $sum = 4 * (4 + 6 + 7 + 8) = 100$.

Răspunsul corect este **b)**.

27. Funcția numără câte elemente din vector au cifrele alăturate de paritate diferită. Ex. *123* par-impar-par, *256* par-impar-par. A se remarca faptul că dacă paritatea setată înainte de eliminarea ultimei cifre (`n/10`) este aceeași cu paritatea actuală, variabila *flag* este setată la *false*.

Răspunsul corect este **d)**.

28. Cele două bucle *for* iterează prin triunghiul superior delimitat de diagonalele principală și secundară. La fiecare linie nouă, celulele sunt populate cu suma a trei elemente de pe rândul precedent, primul rând fiind populat în întregime cu valori de 1. Astfel, pe linia i se asignează doar valori de 3^i, dacă poziția elementului este din triunghiul superior.

Răspunsul corect este **d)**.

29. Funcția *print* parcurge și afișează matricea în spirală, adică prima linie, ultima coloană, ultima linie(ordine inversă), prima coloană(ordine inversă). Apoi, repetă aceeași modalitate de parcurgere pentru ceea ce a rămas din matrice.

Răspunsul corect este **a)**.

30. Bucla *while* are un număr de iterații egal cu dublul dimensiunii matricei pătratice. În cadrul fiecărei iterații vor fi afișate doar elementele care respectă condiția ca suma coordonatelor lor (linie și coloană) să fie egală cu numărul iterației. Astfel, se obține afișarea pseudo diagonalelor secundare și a diagonalei secundare, începând cu cea care pornește din *matrice[0][0]*, urmate de cele care pornesc din *matrice[0][1]*, *matrice[0][2]*, *matrice[0][3]*, ..., *matrice[2][3]* și în final *matrice[3][3]*.

Răspunsul corect este **d)**.

31. Având în vedere modul în care sunt intercalați vectorii și a modului în care este realizată compunerea lor (v depinde de u și u depinde de v) este necesară parcurgerea celor 5 iterații. Acestea sunt următoarele:

iteratia 1: $v = 4\ 0\ 1\ 3\ 2$ $u = 1\ 1\ 0\ 3\ 4$
iteratia 2: $v = 4\ 1\ 1\ 3\ 2$ $u = 1\ 1\ 0\ 3\ 4$
iteratia 3: $v = 4\ 1\ 1\ 3\ 2$ $u = 1\ 1\ 4\ 3\ 4$
iteratia 4: $v = 4\ 1\ 1\ 3\ 2$ $u = 1\ 1\ 4\ 3\ 4$
iteratia 5: $v = 4\ 1\ 1\ 3\ 4$ $u = 1\ 1\ 4\ 3\ 4$

Răspunsul corect este **b)**.

32. Având în vedere modalitatea de accesare al elementelor pe colane (`(i + j) % n`) în loc de obișnuitul index j, putem deduce următorul comportament al programului: prima linie este afișată în ordinea inițială, a doua linie este rotită la dreapta cu 1, a treia linie rotită la dreapta cu 2, etc.

Răspunsul corect este **d)**.

33. Sunt afișate elementele rezultate prin compunerea celor 2 vectori pentru elementele de pe poziții impare, respectiv compunerea cu defazaj de o poziție pentru pozițiile impare.

Răspunsul corect este **a)**.

34. Funcția *print* realizează înmulțirea tuturor elementelor din matrice cu n. A se remarca faptul că toate elementele sunt actualizate, chiar dacă al doilea *for* verifică `j < i` în loc de `j < n`. Acest lucru se datorează faptului că se actualizează atât elementul `m[i][j]`, cât și elementul `m[j][i]`, cele două fiind simetrice față de diagonala principală.

Răspunsul corect este **c)**.

35. Se observă că funcția contorizează elementele din matrice strict mai mari decât elementele aflate în imaginea lor în oglindă față de diagonala principală, fără a lua in calcul elementele de pe diagonala principală. Numărul total al acestor elemente poate fi cel mult $\frac{n(n-1)}{2}$. Acest lucru se întâmplă numai în situația când *fiecare* element aflat deasupra diagonalei principale este mai mare decât elementul corespunzător imaginii lui în oglindă față de diagonala principală. Acest lucru implică faptul că suma tuturor elementele aflate în matrice deasupra diagonalei principale este mai mare decât suma tuturor elementelor aflate în matrice sub diagonala principală.

Răspunsul corect este **b)**.

36. Elementele de pe diagonala principală și de pe diagonala secundară sunt egale cu *'x'*, iar celelalte egale cu *'y'*.

Elementele de pe diagonala principală respecta condiția `i==j`, iar cele de pe diagonala secundară respecta condiția `i+j==n-1`.

Răspunsul corect este **d)**.

37. Matricea c este rezultatul înmulțirii matricelor a și b. Întrucât matricele b și c sunt identice, matricea a trebuie să fie matricea identitate, adică $a = \{\{1,0,0\},\{0,1,0\},\{0,0,1\}\}$.

Răspunsul corect este **c)**.

38. Deasupra diagonalei secundare unde suma indicilor este pară, valoarea atribuită este *'a'*. Sub diagonala secundară unde suma indicilor este impară, valoarea atribuită este *'b'*. În toate celelalte cazuri valoarea este *'1'*.

Răspunsul corect este **c)**.

39. Fiecare element pentru care $i + j$ este impar, va fi egal cu opusul său. Apoi, se interschimbă elementele de pe diagonala principală între ele și elementele de pe diagonala secundară între ele.

Pentru o matrice A = $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$, rezultatul secvenței va fi B = $\begin{pmatrix} d & -c \\ -b & a \end{pmatrix}$.

Răspunsul corect este **d)**.

40. Funcția *f* realizează descompunerea în factori primi ai parametrului *n* și selectează puterea cea mai mare la care este ridicat oricare dintre factorii primi. Programul afișează suma acestor puteri, returnate de funcția *f*. Astfel, se calculează în final $s = 11 + 12 + 8 = 31$.

Răspunsul corect este **a)**.

41. Programul realizează suma tuturor elementelor pare și palindrom de pe liniile indexate impar ale unei matrice.
Pentru linia $\{22, 13, 4, 121\}, index = 0$, par. Nu se modifică s.
Pentru linia $\{32, 44, 18, 23\}, index = 1$, impar. Numerele pare și palindrom: 44.
Pentru linia $\{22, 242, 44, 33\}, index = 2$, par. Nu se modifică s.
Pentru linia $\{11, 1221, 6, 14\}, index = 3$, impar. Numerele pare și palindrom: 6.
În final se calculează $s = 44 + 6 = 50$.

Răspunsul corect este **c)**.

42. Secvența inversează între ele elementele de pe fiecare linie în parte (ultimul inversat cu primul, penultimul cu al doilea, etc.)

Răspunsul corect este **a)**.

43. Programul calculează suma tuturor perechilor de *3* luni consecutive, iar când găsește o sumă mai mare decât maximul deja calculat, stochează în *max_sum* noua valoare. Instrucțiunea `sum = sum + pasi[i][j] - old_value` calculează suma pașilor pe perechea următoare de 3 luni. *pasi[i][j]* reprezintă luna următoare, iar *old_value* luna cea mai din urmă din intervalul curent. *old_value* se calculează ca *p[i][j-3]* dacă *j* este mai mare decât *3*. În caz contrar va trebui să decrementăm *i*-ul (trecem in anul anterior) iar *j*-ul va fi $j + 12 - 3$, deoarece scăzând *3* luni, *j*-ul va deveni negativ și practic vom pleca de indexul maxim al anului anterior. Linia punctată reprezintă exact situația în care avem de calculat un interval care să

Răspunsul corect este **c)**.

44. Se iterează din *2* în *2* de la *0* la *M*, iar pentru fiecare poziție se calculează suma pentru poziția curentă ($Y[i/2][j/2]$) ca fiind:
$X[i][j] + X[i+1][j] + X[i][j+1] + X[i+1][j+1]$ (o matrice de $2x2$)

Răspunsul corect este **a)**.

45. Din ipoteza problemei, se determină valoare lui *m* (numărul de coloane) astfel: Presupunem că valoarea *406* este afișată pe linia cu index *k* și coloana cu index *4* (coloana a *5*-a). Valoarea *402* va fi afișată pe linia *k* și coloana 0.
$402 = k * m$. Factorizând *402*, avem $2 * 3 * 67 = k * m$. Având în vedere condiția impusă asupra lui *m*, rezultă $m = 67$.
De aici, valoarea *872* va fi afișată pe linia *13* și coloana *1*.

Răspunsul corect este **c)**.

46. Expresia din instrucțiunea *if* este adevărată dacă și numai dacă:
- prima cifra a lui *v[i]* nu este identică cu ultima cifră sau
- a doua cifră a lui *v[i]* nu este identică cu penultima cifră sau
- a treia cifră din *v[i]* nu este egală cu antepenultima, adică dacă numărul de *6* cifre *v[i]* nu este palindrom. Dacă *v[i]* este palindrom, este eliminat din vector, adică toate elementele din *v*, de după el, sunt deplasate la stânga cu o poziție.

Răspunsul corect este **c)**.

47. Matricea *mat* este o matrice de rotație cu 90^0 a unui punct în *2D*, în jurul originii. Vectorul *v1* este un punct în plan, care are coordonatele *(1,1)*. Sunt executate *5* rotiri de 90^0 în jurul originii, în sens trigonometric. Rezultă că punctul ajunge în cadranul *2*, la 90^0 față de poziția inițială.
O altă metodă este de a se calcula *5* înmulțiri de matrice.

Răspunsul corect este **a)**.

48. Matricea *mat* este o matrice de rotație cu 90^0 a unui punct în 2D, în jurul originii. Vectorul *v1* este un punct în plan, care are coordonatele *(1,1)*. Sunt executate *1022* rotiri la 90^0 în jurul originii, în sens trigonometric. Rezultă că punctul ajunge în cadranul *3*. Alternativ, se pot face înmulțiri de matrice și se observă că după *4* înmulțiri valorile vectorului *v1* se repetă.

Răspunsul corect este **b)**.

49. Funcția *f* rotește elementele vectorului spre dreapta cu *k* poziții. Pentru cazurile în care *k>n*, atunci se remarcă faptul că după *n* rotiri vectorul revine la forma inițială. Astfel, putem considera că vectorul este rotit cu $k\%n$.

Răspunsul corect este **a)**.

50. Matricea este parcursă de mai multe ori. În prima iterație se atribuie valori întregii matrice. Apoi, la fiecare iterație se parcurge o submatrice obținută prin eliminarea marginilor matricei anterioare. Astfel, asignarea valorilor în matricea *M* trebuie să depindă de variabila *y*.

Răspunsul corect este **c)**.

51. Putem analiza fiecare expresie astfel. Din modul în care este structurat codul se observă că se dorește completarea matricei în funcție de suma $l + c$. Dacă notăm $s = l + c$, atunci $s\%4 == 1$ => asignare z, $s\%4 == 2$=> asignare *y*, iar în celelalte două cazuri, asignare *x*.
`if ((l + c) % 4 == 0 || (l + c) % 4 < 4)` – toată matricea va conține numai *x*, deoarece restul împărțirii la 4 este întotdeauna mai mic decât 4.
`if ((l + c) % 4 == 0)` – pentru *l=0, c=3* nu se face nici o atribuire.
`if ((l + c) % 4 == 3)` – pentru *l=0, c=0* nu se face nici o atribuire.
`if ((l + c) % 4 == 0 || (l + c) % 4 == 3)` – varianta corectă

Răspunsul corect este **d)**.

52. Programul inversează șirul, însă condiția de oprire fiind abia la lungimea șirului, acesta va fi inversat practic de *2* ori. De exemplu, elementele *s[0]* și *s[len-1]* vor fi inversate de *2* ori între ele.

Răspunsul corect este **a)**.

53. Se parcurg toate cele 16 iterații (aferente buclelor for) și se urmărește modul în care sunt atribuite valorile în matricea *A*. Se pot poate analiza și diferențiat pentru a elimina o parte din răspunsuri. De exemplu, pentru $i = 0$ și $j = 1 => A[1][0] = 10$. Astfel, doar răspunsurile *a* sau *c* ar putea fi corecte.

Răspunsul corect este **a)**.

54. Funcția parcurge elementele matricei și contorizează în variabila *nr* numărul acelor elemente din matrice care au proprietatea că au patru vecini în matrice și sunt minime în raport cu toți cei patru vecini. Prima condiție exclude elementele care sunt aflate pe liniile și coloanele de pe marginile matricei. Funcția determină astfel și întoarce numărul elementelor cu această proprietate.

Răspunsul corect este **b)**.

55. Pentru fiecare linie se calculează valoarea elementului maxim, iar cu această valoare se actualizează elementul aflat pe diagonala secundară de pe linia respectivă.

Răspunsul corect este **c)**.

56. Funcția *f* generează matricea

$$mat = \begin{pmatrix} 7 & 7 & 7 & 7 & 7 \\ 6 & 4 & 4 & 4 & 6 \\ 5 & 3 & 3 & 3 & 5 \\ 4 & 2 & 2 & 2 & 4 \\ 3 & 3 & 3 & 3 & 3 \end{pmatrix}$$

Funcția calculează apoi și întoarce suma elementelor de sub diagonala principală din matricea generată *mat. R*ezultă valoarea 34.

Răspunsul corect este **c)**.

57. Primul *strcpy*, obține în *s1* șirul „*Militara Ferdiand I*”.
La al doilea *strcpy*, avem în *s1* șirul „*Militar*”
La al treilea *strcpy*, în *s2* se va obține șirul „*2025*”
Apoi, se concatenează șirurile *s2* la *s1*, obținând în s1 „*Militar2025*”
La final, se concatenează șirul *s2* la *s3*, și obținem în *s3* „*StudentMilitar2025*”.
Răspunsul corect este **c)**.

58. Primul *strcpy* șterge primele *i* elemente ale șirului curent. După bucla *while, cuvant=”reMTA2024”*. În șirul *nou* se copiază *”MTA2024”*.
`nou[strlen(nou) % (strlen(cuvant)+1) - 1] += 1;` schimbă *‘4’* în *‘5’*.
`nou[0] += ('b' - 'B');` schimbă *‘M’* în *‘m’*.
Răspunsul corect este **d)**.

59. Programul calculează radical de ordinul *2* din numărul de caractere al șirului. Dimensiunea șirului fiind de *25*, *aux* va avea valoarea *5*. Se analizează pas cu pas execuția pentru a doua buclă *for* (5 iterații), însă după primul pas răspunsurile sunt deja diferențiate.
Pentru prima iterație, $i = 0$ *și* $aux = 5$, se va afișa *sir[4]*, adică *t*.
Răspunsul corect este **d)**.

60. Programul aduna *k* la fiecare caracter din șirul *șir*. În momentul în care este întâlnit un caracter de spațiu, valoarea variabilei k este incrementată. Astfel, pentru primul cuvânt din *șir* se va aduna valoare 2 la fiecare caracter, pentru al doilea cuvânt valoarea 3 și pentru al treilea valoare 4.
Răspunsul corect este **b)**.

61. Din *sir2* se interschimbă primele două litere, apoi se extrage primul cuvânt din *sir1* (delimitarea fiind dată de spații) și se copiază în *rez*. *(rez=”academia”)*
Se concatenează *sir2* la *rez (rez=”academiatehnica”)*
Apoi, se extrage cel de-al doilea cuvânt din *sir1*, care la rândul său este concatenat la *rez. (rez=”academiatehnicamilitara”)*
În final, caracterul de pe poziția 7 (`strlen(sir2)` fiind 7) este transformat în literă mare *(rez=”academiAtehnicamilitara”)*
Răspunsul corect este **c)**.

62. Programul parcurge fiecare caracter din variabila *sir* și îl afișează în cazul în care este literă mare. Cifrele sunt ignorate. De asemenea, dacă există o tranziție de la literă mare la litera mică sau invers, programul afișează un spațiu *printf(“ “)*. Operația `sir[i]-('a'-'A')` transformă `sir[i]` din literă mică în literă mare. Astfel, se va afișa *ACADEMIA TEHNICA MILITARA*.
Răspunsul corect este **d)**.

63. Funcția *f* parcurge fiecare element al șirului *s*. Dacă acesta este vocală sau caracterul *'_'*, îl șterge, iar dacă este consoană, o transformă în majusculă. Astfel, se va obține un șir care conține doar consoanele șirului inițial, acestea fiind transformate din litere mici în litere mari.

Răspunsul corect este **c)**.

64. Primul *for* va copia a doua jumătate a șirului în prima jumătate și va transforma literele mari în litere mici. *s="mta5202e2025atm".*
Se adaugă terminator de șir pe poziția 7, *s="mta5202".*
Al doilea *for* nu generează nicio modificare asupra șirului, *strlen(s)* fiind mai mic decât 8, *s="mta5atm".*

Răspunsul corect este **c)**.

65. Caracterele din șirul *p1* sunt copiate în șirul *p2*, unul câte unul, după o serie de reguli.
- Caracterul este literă, atunci se copiază valoarea lui – 4 (ex:*,e'* devine *,a'*)
- Caracterul este cifră, se copiază în șir începând cu poziția *c*.
- Caracterul nu este alfanumeric, se copiază la sfârșitul șirului *p2*.

Având în vedere că verificarea pentru cifră se face pentru *p[i]*, nu este corect că copiem *p[i+1]*. Variantele *c* și *d* sunt incorecte.
Pentru a diferenția dintre variantele *a* și *b*, observăm că *'5'* trebuie poziționat pe indexul *11*, prin urmare, avem nevoie de preincrementarea variabilei *c*.

Răspunsul corect este **b)**.

66. Funcția *strrchr* caută ultima apariție a caracterului *,b'* din *str*. Instrucțiunea *str[p1 - str] = '\0';* face ca *str* să devină *„mango,banana,apple,"*. Ulterior, *str* este împărțit în *tokeni* separați prin caracterul *','*. Fiind 3 virgule, obținem 3 *tokeni*.

Răspunsul corect este **d)**.

67. Primul *if* transformă literele mari de pe poziții pare în litere mici. Al doilea *if* înlocuiește literele mici de pe poziții impare cu #. Ramura else elimină caracterele care nu se încadrează în primele două condiții.

Răspunsul corect este **d)**.

68. Funcția *f* parcurge recursiv șirul. Atunci când caracterul curent este par, trece la următorul. Atunci când caracterul curent este impar, îl șterge și apoi reia procesarea de la aceeași poziție. În total, funcția *f* se apelează de *strlen(sir)* ori), astfel: *informatica ->informatica ->iformatica ->iormatica ->iormatica ->iomatica ->iomatica ->iomatica ->iomaica ->iomaica ->iomaica ->iomaica*

Răspunsul corect este **a)**.

69. Primul *if* transformă toate caracterele litere mari în litere mici. Al doilea *if* schimbă caracterele litere mici în litere mari. Al treilea *if* înlocuiește prima apariție a fiecărei vocale cu *'x'*.

Răspunsul corect este **c)**.

70. Programul parcurge șirul *s* de la început până la final, utilizând recursivitatea pentru a ajunge la prima poziție. La fiecare poziție *n*, dacă *n* nu este multiplu de *4*, caracterul de la acea poziție este transformat în majusculă.
Analizăm șirul "ADmitEreATm" și pozițiile fiecărui caracter:
- Poziția 0: *A* (multiplu de 4) — rămâne neschimbat.
- Poziția 1: *D* (nu e multiplu de 4) — rămâne *D* (deja majusculă).
- Poziția 2: *m* (nu e multiplu de 4) — devine *M*.
- Poziția 3: *i* (nu e multiplu de 4) — devine *I*.
- Poziția 4: *t* (multiplu de 4) — rămâne neschimbat.
- Poziția 5: *E* (nu e multiplu de 4) — rămâne *E* (deja majusculă).
- Poziția 6: *r* (nu e multiplu de 4) — devine *R*.
- Poziția 7: *e* (nu e multiplu de 4) — devine *E*.
- Poziția 8: *A* (multiplu de 4) — rămâne neschimbat.
- Poziția 9: *T* (nu e multiplu de 4) — rămâne *T* (deja majusculă).
- Poziția 10: *m* (nu e multiplu de 4) — devine *M*.

Răspunsul corect este **a)**.

71. Programul determină cel mai lung cuvânt din *sir* folosind *j* și *i* ca indici pentru începutul și sfârșitul cuvântului curent. Cuvântul de lungime maximă este stocat folosind indecșii *mi* și *mj*.
Apelul funcției `strncpy(rez, sir + mj, mi)` va copia dir *sir* șirul de dimensiune *mi*, de la poziția *mj* în șirul *rez*.
Obs: Întrucât după ultimul cuvânt nu este un spațiu, acesta nu este luat în considerare în determinarea celui mai lung cuvânt.

Răspunsul corect este **a)**.

72. În cazul în care valoarea caracterului curent este pară, acesta este înlocuit cu simbolul/caracterul '*'. Altfel, este înlocuit cu simbolul/caracterul '#'. Caracterul *'1'* este impar, având corespondent în tabela ASCII valoarea 49.

Răspunsul corect este **a)**.

73. Secvența de program dublează vocalele identice care se găsesc pe poziții simetrice față de mijlocul șirului, punând ulterior caracterul *x* pe vocala dublată.

Răspunsul corect este **d)**.

74. Analizăm fiecare dintre algoritmii menționați, astfel:
- Înmulțirea matricelor are complexitate mai mare de $O(n^2)$ conform Algoritmului Strassen.
- Sortarea vectorilor are în cel mai bun caz complexitate $O(nlog(n))$ dacă se consideră algoritmii *Quick sort* sau *Merge sort*. Pentru *Bucket Sort* sau *Counting sort* complexitatea este $O(n+k)$.
- Căutarea binară are complexitate logaritmică $O(\log(n))$
- Interschimbarea a două variabile are complexitate constantă $O(1)$

Răspunsul corect este **c)**.

75. Primul *for* va itera prin cadrul vectorului *v*, complexitatea generată este $O(n)$. Cele două bucle *for* ulterioare, chiar dacă sunt imbricate, parcurg tot cele *n* elemente. Prin urmare, chiar dacă sunt folosite *2 for*-uri, complexitatea este tot $O(n)$. Complexitatea generală a funcției este $O(n)+O(n)=O(n)$

Răspunsul corect este **a)**.

76. Observăm că funcția recursivă se apelează de *n* ori, iar în fiecare iterație putem considera că se execută $2*k$ operații, unde $k \in [1,n]$. complexitatea programului poate fi calculată folosind:

$$2*\sum_{k=1}^{n} k = 2*\frac{n(n+1)}{2} = O(n^2)$$

Răspunsul corect este **c)**.

77. Numărul total de drumuri este suma drumurilor din celula de deasupra și din celula din stânga. Rezultatul este $C(m+n-2, m-1)$, adică combinările posibile de a alege $m-1$ pași în jos din $m+n-2$. Pentru o matrice $3x3$, se calculează $\binom{4}{2} = 6$.

Răspunsul corect este **b)**.

78. Contorul *i* va lua valorile $\{20,21,22 \dots 2N\}$.
Pentru fiecare valoare a contorului i de tipul $i=2^k$, ciclul for după j execută k iterații (valorile $\{2^0, 2^1, 2^2 \dots 2^{k-1}\}$). Numărul total de iterații (valoarea variabilei *sum*) este deci :

$$1+2+3+\cdots+N = \frac{N(N+1)}{2}$$

Răspunsul corect este **d)**.

79. Funcția calculează frecvența de apariție a caracterelor dintr-un șir de intrare și afișează numărul de apariții pentru fiecare caracter. Varianta c) de răspuns conține șirul potrivit care realizează afișarea menționată.

Răspunsul corect este **c)**.

80. Subprogramul recursiv f generează în vectorul v *ciurul lui Eratostene*: $\begin{cases} v[i] = 0, pentru\ i\ prim \\ v[i] = 1, pentru\ i\ neprim \end{cases}$. Programul afișează toate numele prime mai mici decât 12.

Răspunsul corect este **b)**.

81. Subprogramul calculează numărul de inversiuni dintr-un vector. Inversiunile sunt perechi de elemente (i, j) astfel încât $i < j$ și $v[i] > v[j]$. Pentru exemplul dat, există 5 perechi pentru care cele două condiții sunt satisfăcute simultan.

Răspunsul corect este **b)**.

82. Funcția are scopul de a calcula probabilitatea de a extrage cel puțin *wanted_red* bile roșii în maximum *tries* încercări. Concret apelul `bile(3,3,2,3)` va returna probabilitatea ca extrăgând maximum de bile dintr-o urnă cu 3 bile roșii și 3 bile albastre, voi obține 2 bile roșii.
Probabilitatea se calculează sub formă de raport între numărul de cazuri favorabile și numărul de cazuri posibile. La fiecare apel recursiv se ține cont de bila extrasă la apelul curent prin decrementarea valorilor parametrilor.
În final, probabilitatea este egală cu: $0.5 * 0.6 + 0.5 * 0.3 = 0.45$

Răspunsul corect este **b)**.

83. Analizăm fiecare iterație, considerând ls – limita stânga, ld – limita dreapta și m_i – indexul testat la iterația i.
ls = 1, ld = 9
$m_1 = (ls + ld)/2 = 10/2 = 5$, primul număr testat este $v[5] = 24$
$84 > 24$ astfel că $ls = 6, ld = 9$
$m_2 = 15 / 2 = 7$, al doilea număr testat este $v[7] = 30$
$84 > 30$ astfel că $ls = 8, ld = 9$
$m_3 = 17 / 2 = 8$, al treilea număr testat este $v[8] = 35$
$84 > 35$ astfel că $ls = 9, ld = 9$
$m_4 = 18 / 2 = 8$, al patrulea (și ultimul) număr testat este $v[9] = 80$

Răspunsul corect este **a)**.

84. Subrutina parcurge vectorul folosind indexul *i* pe pozițiile pare, și pentru fiecare element de pe poziție pară testează (folosind iterația după *j*) dacă este diferit de restul elementelor de pe pozițiile pare din vector.

Răspunsul corect este **b)**.

85. Funcția *g* determină cea mai lungă secvență crescătoare a vectorului.

Răspunsul corect este **a)**.

86. Funcția *f* verifică și afișează elementele comune între vectorii *A* și *B*. Fiecare element comun este afișat o singură dată.

Răspunsul corect este **b)**.

87. Se observă faptul că în primul *for* există condiția *i==1*, prin urmare, *for*-urile 2 și 3 se execută o singură dată. A doua și a treia buclă *for* vor genera o complexitate de $O(n^2)$, care va rămâne chiar complexitatea întregii secvențe.

Răspunsul corect este **c)**.

88. Analizăm complexitatea temporală pentru fiecare dintre variante:

a. Complexitate $O(n)$

b. Complexitate $O(n)$

c. Complexitate $O(n^2)$

d. Complexitate $O(nlog(n))$

Cea mai lentă dintre variantele analizate este $O(n^2)$.

Răspunsul corect este **c)**.

89. Algoritmul realizează exponențierea modulară. Cum singurul termen care se modifică este *e*, complexitatea se calculează în funcție de acesta, restul fiind constante. Operația prin care se ajunge la condiția de oprire este împărțirea succesivă la *2* a variabilei *e* => complexitate $O(log_2 e)$.

Putem afirma că oprirea algoritmului este dată de dimensiunea în biți a lui *e*.

Răspunsul corect este **c)**.

90. Analizăm fiecare buclă repetitivă, astfel:

Prima parcurgere - $O(n)$

A doua parcurgere - $O(n)$

A treia parcurgere - *for* are complexitate $O(n)$ iar *while* are $O(\log_4(n))$. Atunci analizăm asimptotic timpul de execuție al unui algoritm, baza logaritmilor poate fi neglijată. A treia parcurgere are complexitate $O(nlog(n))$.

Complexitatea totală a funcției este $MAX(O(n), O\big(nlog(n)\big) = O(nlog(n))$

Răspunsul corect este **b)**.

91. Complexitatea algoritmului lui Euclid este $O(\log n)$, unde n este valoarea maximă dintre a și b. Numărul de pași este proporțional cu numărul de cifre din reprezentarea binară a lui *n*, deoarece resturile scad *repede*.

Răspunsul corect este **b)**.

92. Cele două bucle repetitive sunt imbricate și numărul de iterații nu depinde unul de celălalt (bucla *for* nu depinde de variabila *auxiliar*, folosită de bucla *while*). Astfel, complexitatea programului va fi produsul complexității celor două bucle. Le analizăm pe rând astfel:

- Bucla *for* se apelează de $\sqrt{N}$ ori cu fiecare iterație a blocului *while*.
- Bucla *while* se apelează de $log_3 N$, întrucât *auxiliar* se împarte la 3 la fiecare iterație.

Din punct de vedere al complexității $log_3 N$ și $log_2 N$ sunt echivalente întrucât baza logaritmului se poate schimba printr-o înmulțire cu o constantă:

$$log_3 N = k * log_2 N$$

Răspunsul corect este **d)**.

93. Se analizează pe rând cele două bucle repetitive:

Primul *for* nu se execută deoarece condiția nu se respectă niciodată - $O(1)$

Al doilea set de bucle *for*, poate fi analizat astfel:

for k: se execută de *1000* de ori - $O(1)$

for j: se execută de *n* ori - $O(n)$

for i: se execută de $\log_2 n$ - $O(\log_2 n)$

Complexitatea totală a funcției este:

$$O(1) + O(1) * O(n) * O(\log_2 n) = O(nlog_2 n)$$

Răspunsul corect este **b)**.

94. Pentru determinarea numărului de submulțimi care respectă condiția avem nevoie de utilizarea tehnicii *Backtracking*, complexitatea fiind exponențială (2^n).

Notă: Ar putea exista abordări mai eficiente, bazate pe tehnici precum programarea dinamică însă nici acestea nu ajung la complexitate $O(n)$.

Răspunsul corect este **b)**.

95. Ciclul *for* exterior generează toate numerele întregi $p \geq 2$ a căror rădăcină pătrată este mai mică sau egală cu *n*. Pentru fiecare *p* generat, ciclul *for* interior (imbricat), marchează multiplii lui *p* mai mici decât *n* ca fiind elemente non-prime. Avem $\frac{n}{2} + \frac{n}{3} + \cdots + p = n * \left(\frac{1}{2} + \frac{1}{3} + \cdots\right)$ iterații executate pe baza intrării *n*. Din progresia armonică a sumei numerelor prime rezultă că $\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \cdots = \log(\log(n))$.

Răspunsul corect este **d)**.

96. Funcția *func* se apelează recursiv înjumătățind mereu una dintre valorile *a* și *b*. La fiecare apel, funcția adună *1*. Astfel, deducem că funcția numără de câte ori pot fi împărțiți la 2 parametrii *a* și *b*, însumat.
Cum $8192 = 2^{13}$ *și* $4096 = 2^{12}$, funcția va returna $13 + 12 = 25$ pentru parametrii *8192* și *4096*.

Răspunsul corect este **b)**.

97. Funcția *f* calculează în mod recursiv coeficientul binomial

$$\binom{n}{k} = \binom{7}{3} = \frac{7!}{4! * 3!} = 35.$$

Răspunsul corect este **c)**.

98. Observație: *funct(1) = funct(0) = 0.* Pentru *n* - impar, regăsim o sumă de pătrate pare câte două:
Notând $n = 2k + 1$, obținem $funct(2k + 1) = 2((2k)^2 + (2k - 2)^2 +$ ... $+ 2^2)$. Cum *2* este factor comun și fiecare termen al sumei conține un 2^2, formula se restrânge în $func(n) = func(2k + 1) = 8 * \sum_1^k i^2$.
Cum $\sum_1^k i^2 = \frac{k*(k+1)(2*k+1)}{6}$, $funct(101) = 8 * \sum_1^{50} i^2 = 8 * \frac{50*51*101}{6} =$ *343400*

Răspunsul corect este **a)**.

99. Funcția implementează algoritmul lui Euclid pentru aflarea `cmmdc(a,b)`. Vom avea *ca == 1* când *a* și *b* sunt prime între ele. Vor fi afișate perechile (a, b) prime între ele, generate de apelul recursiv inițializat de la $a = 15, b = 35$. Apelul afișează în ordine următoarele perechi: $(13, 37), (11, 39), (9, 41), (7, 43), (3, 47)$

Răspunsul corect este **b)**.

100. Subprogramul *fun* implementează sortarea recursivă a vectorului *v*, folosind metoda *bubblesort.* Funcția rezolvă o singură iterație a algoritmului, însă vectorul trebuie parcurs de *n* ori. Pe lângă faptul că este nevoie de apelul recursiv al funcției astfel încât să obținem mai multe parcurgeri, este nevoie ca de fiecare dată să fie parcurse cu 1 element mai puțin. Varianta evidentă ar fi fost apelul `fun(v,n-1)` însă aceasta nu se regăsește printre variantele de răspuns. Totuși, având în vedere că $i = n - 1$ atunci când bucla *for* se încheie, putem utiliza *i* în loc de $n - 1$, adică `fun(v,i)`.

Răspunsul corect este **b)**.

101. Funcția calculează în mod recursiv reprezentarea numărului întreg *777* în baza *5* și afișează această reprezentare.

Răspunsul corect este **c)**.

102. Funcția verifică dacă un element al vectorului este prim. Dacă este prim, elementul este adăugat la finalul unui număr construit recursiv. Având în vedere că vectorul este parcurs de la *n* la *0* și a faptului că recursivitatea este apelată înainte compunerii rezultatului, cifrele acestuia sunt ordonate în ordinea în care apar numerele prime în vectorul inițial.

Răspunsul corect este **b)**.

103. Funcția afișează $4 * 100 + 2 + 1$ (ultima ramură *else*, când $n = 0$) + 50 (pentru valorile *n* pare, de la 2 la 100) = 453 caractere * și 50 de caractere # (pentru valorile *n* impare, de la 1 la 100).

Răspunsul corect este **a)**.

104. Funcția *f* calculează recursiv al *n*-lea termen din șirul lui Fibonacci. Funcția *g* caută primul număr din șirul lui Fibonacci care să fie pătrat perfect.

Răspunsul corect este **d)**.

105. Subprogramul *fun* decodifică recursiv un șir de caractere criptat cu un cifru de tip Caesar. În *main*, *fun* este apelată cu cheia *5*. Se decodifică șirul de caractere *v*, adică fiecare caracter se înlocuiește cu caracterul aflat cu *5* poziții în urmă, în alfabet. De exemplu, *R* se înlocuiește cu *M*.

ABCDEFGHIJKL**M**NOPQ**R**STUVWXYZ

Răspunsul corect este **a)**.

106. Programul calculează recursiv suma numerelor *Pell*. Se calculează suma seriei de numere *Pell* pentru *n=11*. Valoarea returnată este mai ușor de calculat pornind de la finalul recursivității, astfel:

$f(0) = 0$
$f(1) = 1$
$f(2) = 2 * f(1) + f(0) = 2$
$f(2) = 2 * f(1) + f(0) = 2 * 1 + 0 = 5$
$f(3) = 2 * f(2) + f(1) = 5 * 2 + 2 = 12$
...
$f(11) = 2 * f(10) + f(9) = 2 * 2378 + 985 = 5741$

Răspunsul corect este **a)**.

107. Se observă că funcția recursivă *getval* implementează de fapt șirul $3, 4, 6, 9, 13, 18, \ldots.$, adică șirul definit de formula $a_n = a_{n-1} + n$, cu $a_0 = 3$
$a_1 = a_0 + 1$, $a_2 = a_1 + 2$, $a_3 = a_2 + 3$, ..., $a_n = a_{n-1} + n$
Se poate observa ușor că $a_n = a_0 + 1 + 2 + \cdots + n = 3 + \frac{n(n+1)}{2}$

$V[2025] = a_{2025} = 3 + \frac{2025*2026)}{2} = 2051328$

Răspunsul corect este **d)**.

108. Funcția recursivă se apelează pentru valorile 28, 27, 26, ... , 10, 9 și returnează suma numerelor ce au ultima cifră mai mare decât 5, penultima cifră pară și au cel puțin 2 cifre. Din șirul menționat, numerele ce respectă cumulativ condițiile sunt: 28, 27 și 26.

Răspunsul corect este **d)**.

109. Analizăm comportamentul funcției variind parametrul *n*, astfel:

$f(0,a,b) = 1$
$f(1,a,b) = a + b$
$f(2,a,b) = a * (a+b) + b * 1$
$f(3,a,b) = a * (a * (a+b) + b) + b * (a+b)$
$= a3 + a2b + 2ab + b2$

Se pot înlocui valorile a, b cu valorile din răspunsuri
Se găsește că f(3,2,1) = 17

Răspunsul corect este **c)**.

110. La fiecare apel de funcție, în variabila s este calculată suma $\sum_{j=1}^{i} j$, iar funcția este apelată de n ori.

Se calculează suma $\sum_{i=1}^{n}\sum_{j=1}^{i} j = \sum_{i=1}^{n} \frac{i(i+1)}{2} = \frac{1}{6}n(n+1)(n+2)$

Răspunsul corect este **a)**.

111. Dacă numărul nu ar fi avut o singură cifră, funcția *f* ar fi returnat chiar valoarea lui *n*. În cazul de față, $n = 7125142$, ca urmare apelul funcției *f* se apelează recursiv și elimină treptat ultima cifră *a* lui *n*. Funcția returnează o valoare construită după cum urmează:

$F(7125142,1) = f(712514,1) * 10 + 2$
$= (f(71251,1) * 10 + 4) * 10 + 2$
$= ((f(7125,1) * 10 + 1) * 10 + 4) * 10 + 2$
$= (((f(712,1) * 10 + 5) * 10 + 1) * 10 + 4) * 10 + 2$
$= (((f(7,1) * 10 + 5) * 10 + 1) * 10 + 4) * 10 + 2$
$= (((7 * 10 + 5) * 10 + 1) * 10 + 4) * 10 + 2$
$= (((750 + 1) * 10 + 4) * 10 + 2) = 75142$

Altfel spus, funcția recursivă descompune și recompune numărul pe cifre. Dacă apare situația în care numărul curent este divizibil cu $c + 1$ și penultima cifră este chiar c, atunci ultimele două cifre sunt eliminate. În acest caz particular, observăm grupul de cifre 12 din număr (care este eliminat).

Răspunsul corect este **b)**.

112. Funcția întoarce suma elementelor din matrice aflate deasupra și pe diagonala secundară, decrementată cu 1. Pentru parcurgerea matricei, funcția folosește un mecanism recursiv de parcurgere pe linii și coloane.

Răspunsul corect este **b)**.

113. Funcția se apelează recursiv eliminând ultima cifra din cadrul numărului întreg *nr*. La fiecare iterație construiește noua valoare *value* astfel, *value = value *10 +* dublul ultimei cifre din numărul *nr* din cadrul iterației respective. Funcția construiește valoarea finală *value* astfel:

$7 + 7 = 14$

$140 + 4 + 4 = 148$

$1480 + 2 + 2 = 1484$

$14840 + 5 + 5 = 14850$

$148500 + 1 + 1 = 148502$

Răspunsul corect este **d)**.

114. Funcția calculează recursiv diferența dintre suma *S1* a elementelor din vectorul v al căror index respectă condiția `index%3==1` sau `index%3==2` și suma *S2* a elementelor din vectorul *v* al căror index respectă condiția *index%3=0* , începând cu indexul 1.

Astfel, $S1 = 61 + 42 + 35 + 74 + 12 + 13 = 237$ și

$S2 = 86 + 68 + 25 = 179$. Elementul 48 din vectorul *v* nu este luat în considerare.

În final, este calculat $S1 - S2 = 58$.

Răspunsul corect este **b)**.

115. Funcția actualizează recursiv coloanele impare în funcție de pragul *p=5*. Dacă valoarea elementului este mai mică decât *5*, elementul este actualizat cu valoarea *-1*.

Astfel, matricea *matrice* modificată este $\begin{pmatrix} -1 & 2 & -1 \\ -1 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$. În final, suma elementelor matricei este 34.

Răspunsul corect este **d)**.

116. Funcția recursivă *f* are următorul rezultat:

$$f(x,y)=\begin{cases}1 & \text{dacă } x=0 \text{ sau } y=0,\\ f(x,y-1)+f(x-1,y) & \text{dacă } y\%2=0,\\ f(x-1,y)+f(x,y-1) & \text{dacă } x\%2\neq 0,\\ 1 & \text{altfel.}\end{cases}$$

Verificăm apelul *f(2,4)*:

$$f(2,4)=f(2,3)+f(1,4)=1+5=6$$
$$f(2,3)=1$$
$$f(1,4)=f(1,3)+f(0,4)=4+1=5$$
$$f(1,3)=f(0,3)+f(1,2)=1+3=4$$
$$f(0,3)=1$$
$$f(1,2)=f(1,1)+f(0,2)=2+1=3$$
$$f(1,1)=f(0,1)+f(1,0)=2$$

Răspunsul corect este **b)**.

117. Se parcurge recursivitatea până când *z* devine mai mare decât *500*. Concret, atunci când $z=800$ recursivitatea se încheie. Toate valorile returnate de apelurile funcției recursive se însumează, astfel: $0+13+14+17+19=63$

Răspunsul corect este **a)**.

118. Apelurile funcției *f* pot fi reprezentate grafic, similar cu un graf. Ulterior, se vor afișa pe ecran nodurile din graf conform unei parcurgeri în adâncime.

Răspunsul corect este **d)**.

119. Programul afișează primele 14 de elemente din șirul lui Fibonacci: 0 1 1 2 3 5 8 13 21 34 55 89 144 233. Analizăm pe rând afirmațiile.
I. Sunt doar 13 elemente distincte deoarece numărul 1 apare de 2 ori, deci afirmația este *falsă*.
II. Șirul este afișat în ordine crescătoare, deci afirmația este *falsă*.
III. 0 este primul element afișat, deci afirmația este *falsă*.
IV. 337 nu este deloc afișat, deci afirmația este *falsă*.

Răspunsul corect este **a)**.

120. Funcția *f* construiește un număr astfel: verifică, prin recursivitate, fiecare cifră a valorii variabilei *a*; de fiecare dată când cifra curentă a lui *a* este egală cu valoarea variabilei *c*, la numărul rezultat se adaugă (prin concatenare) valoarea variabilei *b* urmată de valoarea variabilei *c*. Primul apel generează 414141 iar al doilea apel generează 53.

Răspunsul corect este **c)**.

121. Funcția f returnează 1 dacă cifrele numărului sunt în ordine crescătoare de la stânga la dreapta și 0 în caz contrar.

Răspunsul corect este **a)**.

122. Funcția *g* calculează *cmmdc* între parametrii *a* și *b*. Observăm că funcția *f* extrage ultima și penultima cifră a numărului curent *x* și adună *cmmdc* dintre acestea. Astfel, funcția va returna: $3 + 3 + 4 + 1 = 11$

Răspunsul corect este **c)**.

123. Funcția *func* procesează fiecare cifră a unui număr *n* recursiv, calculând un produs bazat pe proprietățile fiecărei cifre. Pentru fiecare cifră, calculează suma divizorilor proprii ai acesteia. Dacă cifra este pătrat perfect suma calculată anterior este incrementată. Astfel, se va calcula produsul dintre $6 * 3 * 4 * 4 = 288$.

Răspunsul corect este **b)**.

124. Programul verifică dacă un număr *n* este palindrom, folosind funcția recursivă *func* pentru a calcula oglinditul. Dacă acesta este palindrom se va afișa *"DA"*. Pentru ca programul să afișeze de *3* ori *DA*, este necesar să fie introduse *3* numere palindrom.

Răspunsul corect este **a)**.

125. Programul elimină vocalele mici *(a, e, i, o, u)* din șirul de caractere, folosind următorul algoritm: funcția *F* parcurge fiecare caracter din șir; dacă este o vocală o ignoră; dacă nu este vocală, o copiază la o poziție anterioară în șir. Pentru șirul *"AdmitereATM"*, vocalele minuscule sunt eliminate, iar șirul rezultat devine *"AdmtrATM"*.

Răspunsul corect este **c)**.

126. Analizăm ceea ce returnează funcția recursivă:

$$n - x - (x + 3) - (x + 3 + 3) - \ldots - (x + 3 + 3 + \ldots 3 \text{ (de k ori)}) = 0$$

Deci, funcția f întoarce 1 dacă și numai dacă există un *k* pentru care

$$n = x + x + 3 + x + 6 + x + 9 + \ldots + x + 3k$$

$$= x\,(k+1) + 3\,k\frac{k+1}{2} \text{ adică } n = \frac{(k+1)(2x+3k)}{2}$$

Răspunsul corect este **b)**.

127. Observăm că funcția recursivă inițializează vectorul *v* cu următoarele elemente:
$v[0] = 3;$
$v[1] = 5 + 7;$
$v[2] = 9 + 11 + 13;$
$v[3] = 15 + 17 + 19 + 21; etc$
Primul element al vectorului este numărul impar 3, al doilea element este suma următoarelor două numere impare, etc. Primele *29* elemente din vector conțin $1 + 2 + 3 + 4 + \ldots 29 = 29 * 15 = 435$ numere impare consecutive, începând cu numărul impar cu valoarea *3*, adică numerele $2 \cdot 1 + 1, 2 \cdot 2 + 1, 2 \cdot 3 + 1, \ldots . 2 \cdot 435 + 1$. Deci, al 30-lea element din vectorul *v*, are valoarea:

$$V[29] = 873 + + (873 + 2) + (873 + 4) + (873 + 6) + \ldots (\text{de 30 ori})$$
$$= 873 * 30 + 2 * (1 + 2 + 3 + \ldots + 29)$$
$$= 873 * 30 + 29 * 30 = 902 * 30 = 27060$$

Răspunsul corect este **c)**

128. Funcția *init_v* inițializează în mod recursiv vectorii *v_primul* și *v_ultimul*, de *n* elemente cu secvența următoare de valori*:* $(\{1, 3\}, \{4, 7\}, \{7, 11\}, \{10, 15\}, \{13, 19\}, \{16, 23\}, \{19, 27\}, \ldots .$
Perechea a *n*-a din vector este de forma $\{3 \cdot n - 2, 4 \cdot n - 1\}$, Așadar, elementul al *2018*-lea din fiecare vector este *v_primul[2017]* și *v_ultimul[2017]* și are valoarea $\{3 \cdot 2018 - 2, 4 \cdot 2018 - 1\} = \{6052, 8071\}$.

Răspunsul corect este **a)**.

129. Analizăm comportamentul funcției recursive, astfel:
Apelul *f(v, i, j)* calculează în mod recursiv $vi2 + vi + 12 + vi + 22 + vi + 32 + \ldots + vj2$
Apelul `f(v, 0, n-1)` calculează în mod recursiv $vi2 + vi + 12 + vi + 22 + vi + 32 + \ldots + vn - 12$

Apelul *init(v, 200)* inițializează vectorul *v* cu *200* de elemente având valorile $\{1, 2, 3, 4, \ldots, 200\}$. Apelul *f(v, 0, n-1)* calculează suma:
$1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} = \frac{200 \cdot 201 \cdot 401}{6} = 2686700$

Răspunsul corect este **c)**.

130. Programul folosește funcția *F* pentru a roti la stânga șirul *s* cu *k* poziții, de *c* ori.
La primul apel al funcției *F*, șirul "*AdmitereATM*" este rotit la stânga cu *3* poziții: se mută primele 3 caractere *"Adm"* la final, *"AdmitereATM"* devine *"itereATMAdm"*.
La al doilea apel al funcției *F*, șirul *"itereATMAdm"* este rotit din nou la stânga cu 3 poziții: Se mută primele *3* caractere *"ite"* la final, *"itereATMAdm"* devine *"reATMAdmite"*.
După cele două rotații, șirul final devine *"reATMAdmite"*.
Răspunsul corect este **c)**.

131. Se analizează pe rând fiecare afirmație.
a. [0101010101] și [1010101010] sunt singurele secvențe valide ce au 5 valori de 1 – afirmația este *falsă*
b. Următoarele 3 secvențe valide sunt: [0001000001], [0001000101], [0001010001] – afirmația este *falsă*
c. Dacă ar exista 6 valori de 1, s-ar încălca regula de 2 cifre de 1 consecutive – afirmația este *falsă*
d. [0101010101] și [1010101010] sunt singurele secvențe valide ce au 5 valori de 1 și 5 valori de 0 – afirmația este *adevărată*
Răspunsul corect este **d)**.

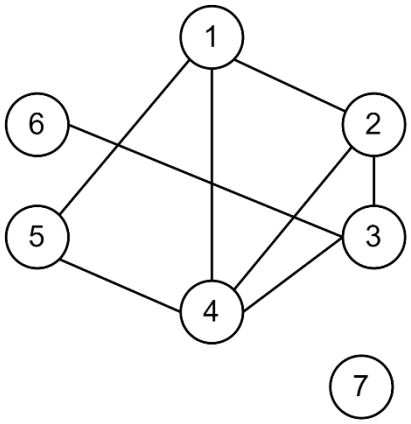
132. Algoritmul folosește backtracking implementat cu funcția recursivă *p* pentru a genera toate permutările șirului. Funcția *p* explorează fiecare posibilitate prin swap-uri, revenind la starea inițială la fiecare pas.
Răspunsul corect este **b)**.

133. Numerele generate sunt: 101 103 105 121 123 125 **141** 143 145 301 303 305 321 323 325 341 343 345 501 503 505 521 523 525 541 **543** 545.
Nu este neapărat necesară generarea tuturor numerelor pentru a găsi răspunsul corect. Pornind de la cele 4 numere din enunț, se generează doar următoarele 3. Referitor la penultimul element, putem considera că 543 este cel corect întrucât după acesta, conform regulilor de generare mai putem genera doar 545.
Răspunsul corect este **a)**.

134. Se continuă generarea de numere respectând ordinea și condițiile din enunț. Numerele generate vor fi: *214, 215, 24, 241*.
Răspunsul corect este **c)**.

135. Observăm că funcția recursivă inițializează vectorul *A* cu cifrele *8* și *9* după următorul șablon: *8, 9, 8, 8, 9, 8, 8, 8, 9, 8, 8, 8, 8, 9, . . .*
La fiecare grup de cifre 8 este inserată în vector o cifra 9.
Dacă presupunem că în vectorul nostru de 55 de cifre, avem un ultim grup complet format din {n cifre de 8 + 1 cifră de 9} iar în plus am putea avea și un grup incomplet, format numai din k cifre 8 $(k < n+1)$, atunci am avea următoarea formulă de calcul:
$V1 + 2 + \ldots +$ n (cifre de 8) $+ 1 + 1 + \ldots + 1$ (cifre de 9) $+$ k (cifre de 8) $= 55$,
adică $\frac{n(n+1)}{2} + n + k = 55$ de unde rezultă $n(n+3) + 2k = 110, cu\ k < n+1$
Soluția acstei ecuații este $n = 9$ și $k = 1$.
Vom avea așadar $k + 1 + 2 + 3 + \cdots + n$ cifre de 8 adică 46 de cifre de 8.
Răspunsul corect este **b)**.

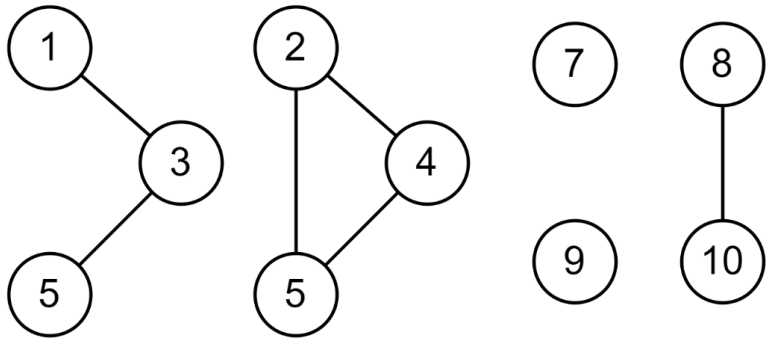
136. Cu toate că nu este neapărat necesară reprezentarea grafică pentru rezolvarea acestei cerințe, pentru ușurința de înțelegere îl reprezentăm astfel:



Se analizează fiecare afirmație, pe rând:
a. Chiar dacă adăugăm o muchie între nodurile 3 și 7, nu putem obține un ciclu hamiltonian – *afirmație falsă.*
b. Numărul de grafuri parțiale admis de un graf neorientat este $2^{număr\ muchii}$. Graful are 8 muchii, deci admite $2^8 = 256$ grafuri parțiale – *afirmație adevărată.*
c. Întrucât graful este neorientat, matricea de adiacență este simetrică față de diagonala principală. Pentru a determina numărul de muchii, putem număra elementele de 1 de deasupra diagonalei principale. Acestea sunt 8. – *afirmație falsă.*
d. Ultimul nod este izolat, prin urmare formează singur o componentă conexă. Astfel, graful are 2 componente conexe – *afirmație falsă*.
Răspunsul corect este **b)**.

137. Fiecare element de pe poziția *i, j* din matricea A^2 arată de câte ori se poate ajunge de la nodul *i* la nodul *j* printr-un drum format din exact două muchii (numărul de drumuri distincte de lungime 2).
Răspunsul corect este **b)**

138. Pentru fiecare linie pară, se anulează valorile de 1 de pe liniile impare. Pentru fiecare linie impară, se anulează valorile de 1 de pe liniile pare.
În urma acestor procesări, reprezentarea grafică a grafului este:



În concluzie, graful are 5 componente conexe.

Răspunsul corect este **a)**

139. Pentru a ajunge la rezultat, vom calcula prima dată numărul total de muchii dintr-un graf cu 8 noduri este $\frac{n(n-1)}{2} = \frac{8(8-1)}{2} = 28$.
Întrucât nodul 5 este izolat, cele 7 muchiile adiacente trebuie eliminate.
Întrucât nodul *2* are graful maxim posibil, dar nodul *5* este izolat, el va avea întotdeauna prezente *6* muchii adiacente. Astfel, problema se reduce la mulțimea grafurilor cu *6* noduri: *{1,3,4,6,7,8}*.
Pentru a putea număra câte grafuri pot fi create, vom considera toate muchiile pe care le putem selecta sau nu. Astfel, avem disponibile $28 - 7 - 6 = 15$ muchii.
A se observa că numărul de 15 muchii putea fi obținut și ca numărul total de muchii al unui graf cu 6 noduri ($\frac{n(n-1)}{2} = \frac{6(6-1)}{2} = 15$).
Numărul total de grafuri care pot fi create cu cele *15* muchii este 2^{15}.

Răspunsul corect este **a)**.

140. Argumentele subprogramului *fun* sunt numărul de noduri (variabila *n*) și indecșii nodurilor din matricea de adiacență (variabilele *i* și *j*). Numărul de noduri este 8, *i* reprezintă liniile matricei de adiacență, iar *j* coloanele. Variabila *i* trebuie să fie *0*, pentru ca subprogramul să parcurgă toate liniile matricei, dar *j* poate fi *1* întrucât prima valoare din matricea de adiacență exprimă legătura primului nod cu el însuși.

Răspunsul corect este **b)**.

141. În acest caz, informațiile furnizate nu sunt suficiente pentru a determina ceva sigur privitor la faptul că graful este Eulerian sau Hamiltonian.
Numărul de componente conexe este 3 (poate fi reprezentat grafic graful).
Cum toate gradele nodurilor sunt numere pare, atunci cel puțin o componentă conexă a sa definește un graf Eulerian.

Răspunsul corect este **d)**.

142. Analizăm pe rând fiecare variantă, astfel:
- varianta *a*: conține noduri terminale, deci graful nu poate fi hamiltonian.
- varianta *b*: conține un nod izolat, deci graful nu poate fi hamiltonian.
- varianta *c*: poate fi asociată unui graf hamiltonian. Toate nodurile au gradul mai mare decât 1 (condiție necesară, dar nu suficientă).
- varianta *d*: conține noduri terminale, deci graful nu poate fi hamiltonian.

Un exemplu de graf pentru varianta c poate fi cel definit de matricea de adiacență:

$$\begin{pmatrix} 0&1&1&0&0&0 \\ 1&0&0&1&1&1 \\ 1&0&0&0&1&0 \\ 0&1&0&0&0&1 \\ 0&1&1&0&0&1 \\ 0&1&0&1&1&0 \end{pmatrix}$$

Răspunsul corect este **c)**.

143. Programul verifică dacă toate nodurile grafului au gradul număr prim și mai mare decât 0. Analizăm fiecare variantă în parte astfel:
- varianta *a*: conține un nod (*3*) cu gradul egal cu 1 deci nu este prim.
- varianta *b*: conține un nod (*3*) cu gradul egal cu 1 deci nu este prim.
- varianta *c*: conține un nod cu gradul 0, deci nu respectă a doua condiție.
- varianta *d*: îndeplinește condițiile având numai noduri de grade 2 și 3 (prime).

Răspunsul corect este **d)**.

144. Numărul de muchii într-un graf complet $\frac{n(n-1)}{2} = \frac{7*6}{2} = 21$

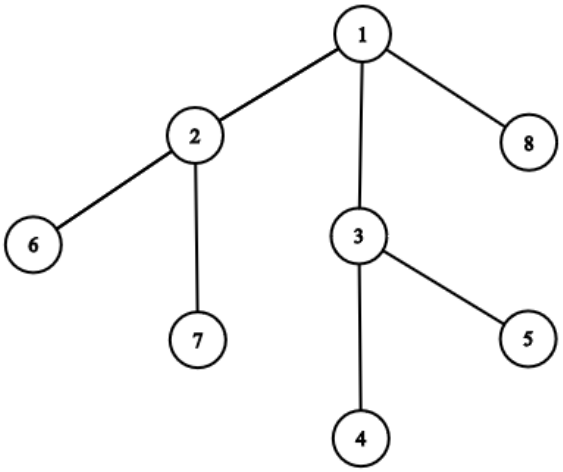
Numărul grafurilor parțiale se obține prin numărarea tuturor posibilităților de a selecta sau nu muchiile din graf. Astfel, numărul de grafuri parțiale pentru un graf cu *M* muchii este 2^M.

Răspunsul corect este **c)**.

145. Singura variantă ca un arbore binar să conțină o singură frunză este cea în care fiecare nod are exact un fiu. Înălțimea arborelui contorizează numărul de muchii, de la rădăcină până la frunză.

Răspunsul corect este **a)**.

146. Graful poate fi reprezentat grafic astfel:



Din figură remarcăm faptul că graful nu conține cicluri și că este conex.

Răspunsul corect este **a)**.

147. Analizăm fiecare afirmație în parte, astfel:

- Modul de selecție al muchiilor nu permite cicluri – *afirmație falsă*
- Graful are cel puțin *1000* de muchii deoarece nodurile x de la *1* la *1014* formează perechile, $(x, x + 1010)$, aceste perechi fiind exact *1014 – afirmație adevărată*
- Noduri precum 5 și 1015 au doar o muchie între ele, prin urmare constituie o componentă conexă, separată de restul grafului – *afirmație falsă*
- Cel mai mare grad al grafului îl au nodurile *1011*, *1012*, *1013* și *1014*, cu gradul *2*, acestea având *2* muchii, ex: *(1,1011)* și *(1011, 2021) – afirmație falsă*

Răspunsul corect este **b)**.

148. Analizăm fiecare dintre afirmații astfel:

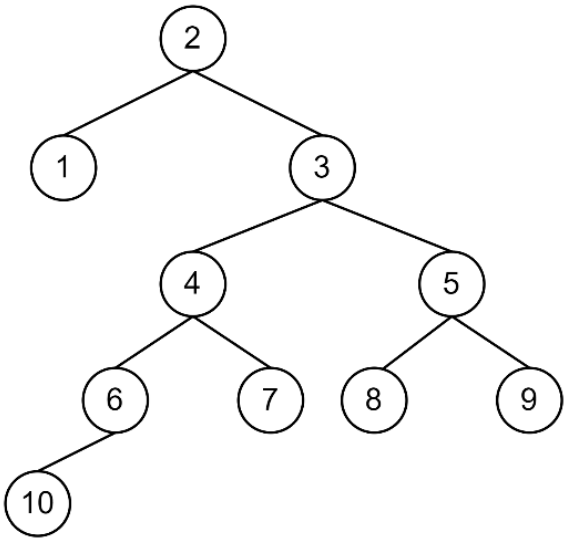
- graful este hamiltonian. Orice graf complet cu mai mult de 3 noduri este hamiltonian. *Afirmație adevărată.*
- gradul este eulerian. Graful fiind complet, nodurile au gradul 99. Pentru a fi eulerian, toate gradele ar fi trebuit să fie pare. *Afirmație falsă.*
- graful are 4950 muchii. Fiind complet, graful conține $1+2+\ldots+99 = \frac{99*100}{2} = 4950$. *Afirmație adevărată.*
- dacă se elimină 50 de muchii, graful poate deveni eulerian. Dacă sunt eliminate 50 de muchii, între noduri consecutive, adică 1-2, 3-4, 5-6, etc. Fiecare nod va avea gradul 98. *Afirmație adevărată.*

Răspunsul corect este **b)**.

149. Analizăm fiecare proprietate de interes a grafului, astfel:
- Suma gradelor este egală cu $2 * numărul\ de\ muchii = 14$.
- Graful este conex.
- Un graf neorientat conex nu poate conține un ciclu eulerian dacă există noduri cu grade impare. Nodurile *e* și *f* au grad impar.

Răspunsul corect este **b)**.

150. Pentru o bună vizualizare, se reprezintă grafic arborele astfel:



Înălțimea actuală a arborelui este de *4*. Dacă ar fi ales ca rădăcină nodul *3* sau nodul *4*, atunci înălțimea arborelui ar deveni *3*, care este și înălțimea minimă posibilă.

Răspunsul corect este **b)**.

151. Numărul de noduri poate fi calculat astfel:

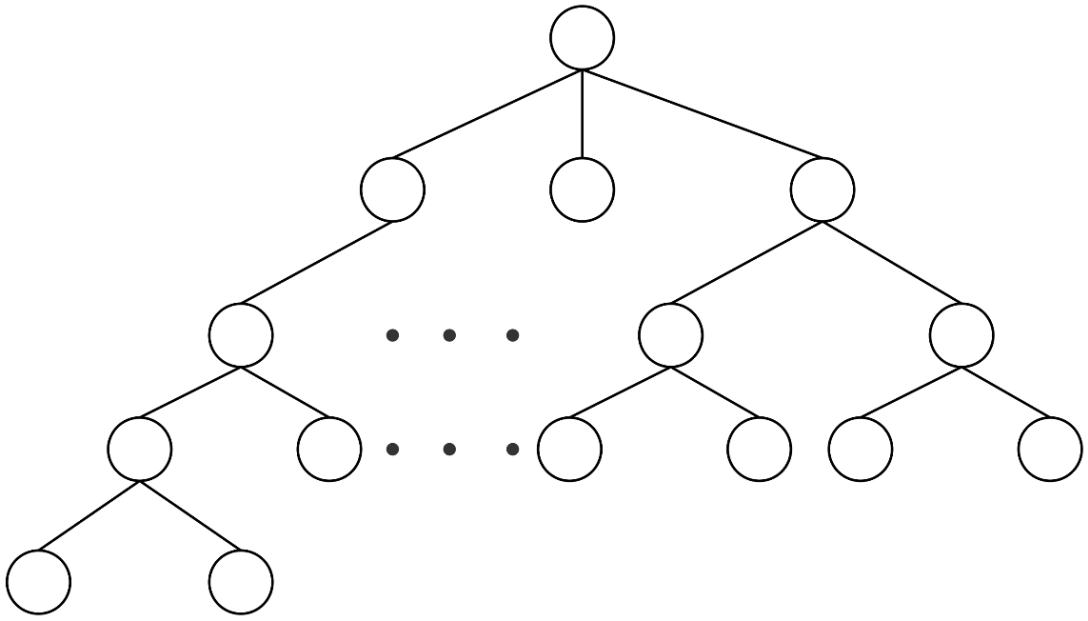
$$\sum_{i=0}^{h} 2^{n*i} = 1 + 2^n + 2^{2n} + 2^{hn}$$

Observăm că avem de calculat suma unei progresii geometrice, astfel:

$$N = \frac{a(r^{k+1}-1)}{r-1} \ unde \begin{cases} a = 1 \\ r = 2^n \\ k = h \end{cases} \text{ adică } N = \frac{2^{n(k+1)}-1}{2^n-1}.$$

Răspunsul corect este **a)**.

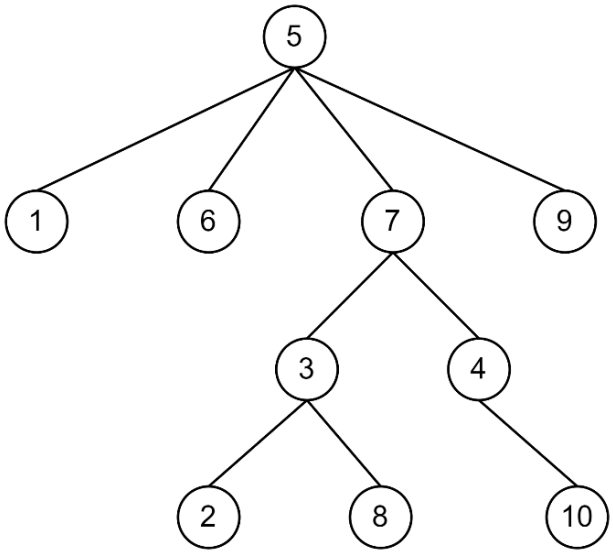
152. Arborele care poate respecta cerința are 3 fii ai rădăcinii, după care fiecare nod are exact 2 fii. În acest mod, fiecare nod are gradul 3. Frunzele au gradul 1. Structura arborelui poate fi observată în figura următoare:



Numărul de noduri este următorul: $1 + 3 * 2^1 + 3 * 2^2 + 3 * 2^3 + \cdots$
Putem observa că pentru un arbore cu înălțime 3, numărul de noduri este:
$1 + 3 * 2^1 + 3 * 2^2 + 3 * 2^3 = 1 + 3 + 6 + 12 = 22$
Astfel, este nevoie de încă 2 frunze pentru a completa ultimul nivel. Lanțul elementar cel mai lung va porni de la frunza de pe nivelul 4, prin rădăcină, până la cea mai îndepărtată frunză din cealaltă parte a arborelui.

Răspunsul corect este **a)**.

153. Arborele poate fi reprezentat grafic astfel:



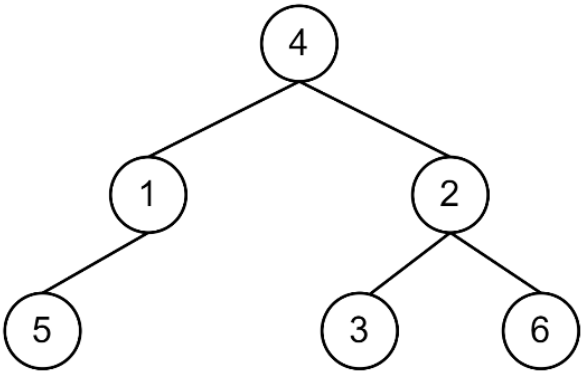
Se observă că se pot forma mai multe lanțuri elementare de lungime maximă, precum {2, 3, 7, 4, 10}, {8, 3, 7, 4, 10}, lungimea acestora fiind 4.

Răspunsul corect este **a)**.

154. Funcția procesează iterativ fiecare nod în parte. Pentru fiecare astfel de nod parcurge vectorul de tați până când ajunge la rădăcină. Procesul implică și înregistrarea distanței parcurse, în variabila *cale*. În final, programul reține în *maxim* calea cea mai mare și în variabila *nod*, nodul cel mai îndepărtat de rădăcină. Se returnează *maxim*, acesta fiind distanța de la rădăcină la cel mai îndepărtat nod, adică înălțimea arborelui.

Răspunsul corect este **d)**.

155. Pentru nodul 4 ca rădăcină arborele poate fi reprezentat astfel:



Înălțimea arborelui este 2. Nu poate fi reprezentat un alt arbore cu o înălțime mai mică, prin simpla schimbarea nodului rădăcină și păstrarea muchiilor.

Răspunsul corect este **a)**.

COLECȚIE DE CHESTIONARE PENTRU ADMITERE

IULIE 2021

1. Indicați valorile variabilelor **x** și **y** obținute în urma execuției următoarelor instrucțiuni scrise în limbajul C:

```
int x, y;
x = 2; y = 4;
y = y - 1;
x = x + y;
```

a) x = 5, y = 3; b) x = 5, y = 4; c) x = 5, y = 6; d) x = 6, y = 3; e) x = 7, y = 3.

2. Care dintre următoarele opțiuni conține doar tipuri de date ce se găsesc în limbajul C?

a) int, *char, float
b) int, char, double, triple
c) int, char, float, double, long int
d) int, char, NULL, float
e) char, float, int[5], blob

3. Indicați valoarea variabilei **x** în urma execuției următoarelor două linii de cod C:

```
int x = 14;
x = x % 3 + x / 6 + 5;
```

a) 5; b) 7; c) 6; d) 7.5; e) 9

4. Precizați valoarea constantei **a** pentru care următoarea secvență de cod C va afișa pe ecran exact mesajul Admitere_ATMFI:

```
int x = 2021;
while (x > a) {
        x = x - 1;
        if (x % 4 != 0) printf("ATMFI");
        else printf("Admitere_");
}
```

a) 2018; b) 2019; c) 2020; d) 2021; e) 2022

5. Indicați valoarea variabilei **x** după execuția următoarei secvențe de instrucțiuni:

```
int x = 3, i;
if (x < 2)
        for (i = x; i < 50 * x; i *= 2)
                x += i;
```

a) –3; b) 500; c) 0; d) 3; e) 3503

6. Matricea pătratică **B** având **n** × **n** elemente de tip întreg (cu liniile și coloanele indexate de la **0** la **n–1**) se obține din matricea pătratică **A** prin rotirea matricei **A** spre stânga cu 90° (în sensul invers al acelor de ceasornic).

De exemplu, dacă matricea **A** este $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$,

atunci matricea **B** obținută în urma rotirii este: $\begin{pmatrix} 4 & 8 & 12 & 16 \\ 3 & 7 & 11 & 15 \\ 2 & 6 & 10 & 14 \\ 1 & 5 & 9 & 13 \end{pmatrix}$.

Menționați linia de program care trebuie să fie completată în locul liniei punctate din cadrul secvenței de cod precizată mai jos pentru a realiza în mod corect generarea matricei **B**:

```
for (i = 0; i< n; i++)
        for (j = 0; j < n; j++)
                ..... ;
```

a) B[i][j] = A[j][i]

b) B[i][j] = A[n - j - 1][n - i - 1]

c) B[i][j] = A[n - j - 1][i]

d) B[i][j] = A[n - i - 1][j]

e) B[i][j] = A[j][n - i - 1]

7. Se consideră un graf orientat cu 5 noduri, reprezentat prin listele de vecini prezentate în tabelul următor:

Nod	Listă de vecini
1	2, 3
2	1, 5
3	2, 4
4	3
5	1, 4

Câte muchii rămân în graf după eliminarea nodului cu gradul extern (de ieșire) minim?

a) 9; b) 4; c) 6; d) 5; e) 7

8. Indicați care este varianta corectă de utilizat în locul spațiului punctat, astfel încât secvența de cod de mai jos să afișeze toate numerele pare aflate în intervalul [100,1000].

```
for (.....)
        printf("%d ", i / 4);
```

a) i = 400; i <= 4000; i += 8
b) i = 100; i <= 1000; i += 2
c) i = 400; i < 1400; i++
d) i = 100; i < 1000; i++
e) i = 800; i <= 8000; i += 8

9. Pentru subprogramul f, definit mai jos, indicați valoarea obținută în urma apelului f(7,5).

```
int f(int a, int b)
{
    if (a <= 0)
        return 0;
    return f(b - 1, a - 1) + a - b;
}
```

a) 3; b) 14; c) –10; d) 2; e) 0

IULIE 2022

1. Precizați ce complexitate are algoritmul implementat de funcția *func*. Parametrii cu care funcția este apelată sunt: v un vector de elemente întregi, *N* numărul de elemente ale vectorului *v*, iar *i* un număr întreg cu valoarea 0.

```
int func(int v[ ], int N, int i)
{
        if (i >= N) return 200;
        return v[N - i - 1] + func(v, N, i + 1);
}
```

a) $O(N)$; b) $O(\log(N))$; c) $O(N!)$; d) $O(N^2)$; e) $O(\frac{1}{N})$.

2. Precizați ce se afișează la apelul funcției *myprint* definită mai jos, dacă este folosită într-un program C pentru un număr natural strict pozitiv *X:*

```
void myprint(int X)
{
        if (X) {
                myprint(X / 2);
                printf("%d", X % 2);
        }
}
```

a) 1 dacă numărul *X* este par, respectiv 0 dacă numărul *X* este impar;
b) 0 dacă numărul *X* este par, respectiv 1 dacă numărul *X* este impar;
c) 1 dacă numărul *X* este o putere a lui 2;
d) Cifrele numărului *X* scris în baza 2;
e) Cifrele numărului *X* scris în baza 2, dar afișate în ordine inversă.

3. Se dă graful neorientat definit de nodurile $V = \{ 2, 3, \dots, 2022 \}$ și setul de muchii $E = \{(x, x^2), \text{astfel încât } x \in V \text{ și } x^2 \in V\}$. Care propoziție este adevărată?

a) Graful este conex;
b) Graful are un ciclu;
c) Graful are 43 de muchii;
d) Gradul cel mai mare al unui nod este 4;
e) Toate răspunsurile de mai sus sunt corecte.

4. Un arbore cu 6 noduri, numerotate de la 1 la 6, are drept rădăcină nodul numerotat cu 2 și muchiile: [1,3], [2,6], [2,1], [6,5], [5,4]. În acest arbore mai sunt adăugate ulterior două noduri: 7 și 8. Care este numărul maxim de noduri de tip frunză pe care le poate avea arborele obținut?

a) 1; b) 2; c) 3; d) 5; e) 4.

5. Precizați care este structura tabloului bidimensional *A* după execuția următoarei secvențe de program C:

```
int A[3][3];
int i, j;
for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
                if ((i + j) % 3 == 0) A[i][j] = 1;
                else A[i][j] = 0;
```

a) $\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}$; b) $\begin{pmatrix}0&1&1\\1&1&0\\1&0&1\end{pmatrix}$; c) $\begin{pmatrix}1&1&1\\1&1&0\\1&0&0\end{pmatrix}$; d) $\begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix}$; e) $\begin{pmatrix}0&0&0\\0&0&1\\0&1&1\end{pmatrix}$.

6. Se consideră funcția *myprint* definită mai jos. Parametrul *v* este un vector de elemente întregi, iar parametrul *n* reprezintă numărul de elemente din cadrul vectorului *v*.

```
void myprint(int v[ ], int n)
{
        int i;
        for (i = 0; i < n; i++)
        {
                if (i % 2)
                        if (v[i] % 2 == 1)
                                printf("%d  ", v[i]);
        }
}
```

Apelarea funcției *myprint* va afișa:

a) Toate elementele pare din cadrul vectorului;

b) Toate elementele impare din cadrul vectorului;

c) Elementele impare din cadrul vectorului, aflate pe poziții impare;

d) Elementele pare din cadrul vectorului, aflate pe poziții impare;

e) Elementele impare din cadrul vectorului, aflate pe poziții pare.

7. Indicați valoarea variabilei *x* după execuția următoarei secvențe de instrucțiuni:

```
int x = 3;
/* x = x--; */
x += 1;
x = x / 2;
```

a) 2; b) 1; c) 3; d) 0; e) 5.

8. Precizați ce se afișează la execuția programului definit mai jos:

```
void main()  {
        int n = 20021, val = 0;
        int i, j, m, ok;
        for (i = 1; i <= n; i++)
        {
                m = n; j = i; ok = 1;
                while (j > 0 && ok == 1)
                {
                        while (m > 0)
                        {
                                if (m % 10 == j % 10) break;
                                m = m / 10;
                        }
                        if (m == 0) {
                                ok = 0; break;
                        }
                        m = m / 10;
                        j = j / 10;
                }
                if (ok == 1) val++;
        }
        printf("%d", val);
}
```

a) 0; b) 5; c) 13; d) 17; e) 2021.

9. Precizați ce valoare returnează apelul funcției *f(1239547632)*, unde funcția *f* este definită mai jos:

```
long f(long n)
{
        long k, x;
        if (n == 0) return 0;
        else
        {
                x = ((n / 10) % 10) * 10 + n % 10;
                for (k = 2; k < x; k++)
                        if (x % k >= 1) continue;
                        else break;

                if (k < x) return f (n / 10);
                else return f (n / 10) * 10 + n % 10;
        }
}
```

a) 1239547632; b) 12347; c) 139573; d) 1; e) 137.

IULIE 2023

1. Indicați rezultatul execuției programului prezentat mai jos:

```
int to_print(int n){
      if (n%2==0 && n%3==0)
            return 1;

      return 0;
}
void main() {
      int i;
      for (i = 1; i <= 100; i++) {
            if (to_print(i) == 1)
                  printf("%d ", i);
      }
}
```

a. Afișarea tuturor numerelor divizibile cu 2 din intervalul (1,100)
b. Afișarea tuturor numerelor prime din intervalul (1,100)
c. Afișarea tuturor numerelor divizibile cu 6 din intervalul [1,100]
d. Afișarea tuturor numerelor divizibile cu 3 din intervalul [1,100]

2. Fie dată matricea pătratică *matrix*, având *n* linii și *n* coloane, indexate între *0* și *n-1*. Alegeți varianta corectă pentru a înlocui linia punctată astfel încât secvența de instrucțiuni obținută să afișeze toate elementele de pe diagonala secundară și de sub diagonala secundară.

```
for (i=0; i<n; i++) {
      ..............
            printf("%d ", matrix[i][n-j-1]);
}
```

a. `for (j = i; j >= 0; j--)`
b. `for (j = i; j > 0;  j--)`
c. `for (j = 0; j < i;  j--)`
d. `for (j = 0; j < i;  j++)`

3. Fie două variabile întregi *a* și *b*. Valorile stocate în aceste variabile sunt cuprinse în intervalul [0,999]. Secvența de cod de mai jos are ca scop interschimbarea valorilor variabilelor *a* și *b* fără a utiliza o variabilă suplimentară.

```
a *= 1000;
.........
b = (a-b)/1000;
a -= (b*1000);
```

Cu ce trebuie înlocuită linia marcată prin ..... pentru ca secvența de cod să îndeplinească în mod corect cerința propusă?

a. `a -= b;`
b. `a += b;`
c. `b += a;`
d. `b -= a;`

4. Ce va fi afișat în urma execuției programului definit mai jos?

```
void f(char s[], int i, int n){
      if (i >= n-1)
            return;
      if ((s[i]>='a' && s[i]<='z') || (s[i]>='A' && s[i]<='Z'))
            s[i] = '*';
      else
            s[i] = '$';
      f(s, ++i, n);
}
void main(){
      char sir[] = "paro22la";
      f(sir, 0, sizeof(sir));
      f(sir, 0, sizeof(sir));
      printf("sir:%s|", sir);
}
```

a. sir:****$$**|
b. sir:********|
c. sir:*$*|
d. sir:$$$$$$$$|

5. Precizați ce se afișează în urma execuției programului de mai jos:

```
int f(int x) {
      return x/2023 - x%2023 + 2023;
}
void main() {
      int k, val=f(0);
      for (k=1; k<=9999; k++)
            if (val<f(k)) val=f(k);
      printf("%d", val);
}
```

a. 2027
b. 9999
c. 2025
d. 2023

6. Se consideră funcția de mai jos, unde *v* reprezintă un vector cu elemente întregi sortate crescător. Funcția *search* își propune căutarea indexului la care se află elementul *target* în vector. Cu ce trebuie înlocuită secvența ..... pentru ca funcția să își îndeplinească în mod corect obiectivul?

```
int search(int v[], int left, int right, int target) {
      int mid = left + (right - left) / 2;
      if (right >= left) {
            if (v[mid] == target)
                  return mid;
            if (v[mid] > target)
                  return search(v, left, mid - 1, target);
            else
                  return ................... ;
      }
      else
            return -1; /*nu este gasit elementul*/
}
```

a. `search(v, 0,     mid,   target)`
b. `search(v, right, mid,   target)`
c. `search(v, mid+1, right, target)`
d. `search(v, left,  right, target)`

7. Precizați prima valoare afișată la apelul *f(51,51):*

```
int f(int n, int k) {
      int i, start;
      if (n==1) {
            if (n==k) printf("(%d)", 0);
            return 0;
      }
      start = f(n-1, k) + 1;
      for (i = start; i < (start+n); i++) {
            if (n==k) printf("%d", i);
            if (i < start+n-1)
                  if (n==k) printf(",");
      }
      return i-1;
}
```

a. 1539
b. 1275
c. 1326
d. 1034

8. Fie un graf neorientat dat prin matricea de adiacență de mai jos:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Care dintre afirmațiile de mai jos este corectă?

a. Graful poate fi un arbore cu 3 frunze
b. Graful este complet
c. Graful nu este conex
d. Graful este un arbore cu 6 frunze

9. Câte grafuri neorientate, distincte, cu 6 vârfuri se pot construi?
Două grafuri se consideră distincte dacă matricele lor de adiacență sunt diferite.
În matricea de adiacență a unui graf neorientat, elementele de pe diagonala principală sunt nule.

a. 2^{36}
b. 2^{21}
c. 2^{10}
d. 2^{15}

SEPTEMBRIE 2023

1. Funcția de mai jos trebuie să returneze maximul a 3 valori întregi, primite prin parametrii `a`, `b` respectiv `c`.

```
int maximum(int a, int b, int c){
     if (a > b)
     {
          if (a > c)
               return a;
     }
     ..............
          return c;
     return b;
}
```

Cu ce trebuie înlocuită linia marcată prin `..............` pentru ca funcţia să îndeplinească în mod corect sarcina propusă?

e. `if (a > c)`
f. `if (b < c)`
g. `if (a < c)`
h. `if (c > a)`

2. Un arbore cu 7 noduri, numerotate de la 1 la 7, are vectorul de tați (3,3,0,1,1,5,6). Care este nodul de tip frunză cu numărul cel mai mic?

a. 7
b. 3
c. 4
d. 2

3. Indicați valoarea variabilei c după execuția următoarei secvențe de instrucțiuni:

```
char a, b, c;
a = 'A';
b = ++a;
c = (a + b) % 2;
printf("%d",c);
```

a. 1
b. 2
c. 0
d. 65

4. Un şir de caractere este „util” într-un program dacă are lungimea de 7 caractere şi conţine numai caractere din mulţimea $\{A, B\}$. Precizaţi care este al 90 –lea şir „util” de caractere, în ordine alfabetică?

a. BABBAAB
b. BAABBBA
c. ABBBAAB
d. BABABAA

5. Variabila A memorează un tablou bidimensional cu 10 linii și 10 coloane, având inițial toate valorile elementelor egale cu 0. Precizați care este suma valorilor elementelor din tabloul A, după executarea funcției de mai jos.

```
void f(int A[10][10]){
  int x, y;
  for (x = 1; x <= 8; x++)
       for (y = 1; y <= 8; y++)
            if (x % 2 == 0)
                 A[x][y] = (x - 1) % 4;
            else
                 A[y][x] = y - 1; /*atentie la indexare*/
}
```

a. 176
b. 144
c. 108
d. 116

6. Funcția de mai jos trebuie să returneze primul index la care se află valoarea val în șirul sir, sau -1 dacă valoarea căutată nu este găsită.

```
int linSearch(int sir[], int dim, int val){
    int i;
    for (i = 0; i < dim; i++)
    {
        if (sir[i] == val)
            ............
    }
    return -1;
}
```

Cu ce trebuie înlocuită linia marcată prin `............` pentru ca funcţia să îndeplinească în mod corect sarcina propusă?

a. `break;`
b. `return val;`
c. `return i;`
d. `continue;`

7. Pentru subprogramul definit mai jos, indicați valoarea returnată pentru apelul $function(3,25)$.

```
int function(int a, int b){
      int x = 1, i;
      for (i = a; i < b; i += x){
            if (i % 3 == 2){
                  x += a / 2;
                  a++;
            }
            x++;
            b = b + 2;
      }
      return x;
}
```

a. 9
b. -11
c. 10
d. 23

8. Indicați valorile afișate pe ecran după execuția secvenței de cod de mai jos:

```
int i = 0, j = 0, a = 0;
int M[4][4] = {0};

for (i = 0; i < 16; ++i){
    M[i / 4][i % 4] = i;
}

for (i = 0; i < 4; ++i) {
    a = M[i][i];
    M[i][i] = M[i][4 - i - 1];
    M[i][4 - i - 1] = a;
}

for (i = 0; i < 4; ++i){
    printf(" %d", M[i][i]);
}
```

a. 3 6 9 12
b. 0 5 10 15
c. 4 5 6 7
d. 0 4 8 12

9. Funcția definită mai jos implementează tehnica backtracking. Se presupune că vectorul v este format din elemente distincte, iar ambii vectori au suficient spațiu de memorie alocat. Funcția se apelează cu parametrii $step = 0$ și n dimensiunea vectorului v. Care este rezultatul obținut la apelul funcției $backtrack$?

```
int backtrack(int v[], int vout[], int step, int n)
{
	int i, j, sw;
	if (step == n)
		return 1;
	for (i = 0; i < n; i++)
	{
		vout[step] = v[i];
		if (step > 0 && vout[step - 1] > vout[step])
			continue;
		sw = 0;
		for (j = 0; j < step; j++)
			if (vout[step] == vout[j])
				sw = 1;
		if(sw)
			continue;
		if (backtrack(v, vout, step + 1, n))
			return 1;
	}
	return 0;
}
```

a. Generarea vectorului $vout$ ca o copie a lui $vout$
b. Generarea vectorului $vout$ din elementele lui v, sortate crescător
c. Generarea vectorului $vout$ din elementele lui v, sortate descrescător
d. Generarea vectorului $vout$ ca o copie a lui v

IULIE 2024

1. Indicați valoarea afișată pe ecran după execuția programului următor.

```
void main(){
	int i = 0, c = 23, d = 51;
	d = i || c;
	i = d && c;
	c = !(d && i);
	printf("%d", c);
}
```

a. 1
b. 0
c. 89
d. 23

2. Fie *n* un număr întreg din intervalul [1, 10000]. Precizați ce returnează funcția *func*.

```
int func(int n){
	int digit, c = 0;
	while (n > 0)
	{
		digit = n % 10;
		c += digit % 2;
		n = n / 10;
	}
	return c;
}
```

a. Litera "c"
b. Numărul de cifre impare al parametrului *n*
c. Numărul de cifre al parametrului *n*
d. Numărul de cifre pare al parametrului *n*

3. Precizați ce se va afișa în urma execuției programului următor:

```
void f(int A[], int n){
	int i, x;
	for (i = 0; i < n / 2; i++)
	{
		x = A[i];
		A[i] = A[n - 1 - i];
		A[n - 1 - i] = x;
	}
	for (i = 0; i < n; i++)
		printf("%d ", A[i]);
}
void main(){
	int A[] = { 1,2,3,4,5,6,7,8,9,10,11,12,13,14 };
	f(A, 8);
}
```

a. 8 7 6 5 4 3 2 1
b. 7 6 5 4 3 2 1 8
c. 9 10 11 12 13 14 15 16
d. 14 15 13 12 11 10 9 8

4. Un erou poartă o *armură* și o *cască*. Materialele din care acestea pot fi confecționate pot fi: *bronz*, *argint*, *aur*, *platină*. Purtarea concomitentă a unui obiect de *bronz* cu un obiect de *platină* este interzisă. Utilizând metoda backtracking, este generată lista completă de combinații pentru materialele din care pot fi fabricate cele două obiecte. Primele 5 combinații generate sunt:

cască – bronz, armură - bronz
cască – bronz, armură - argint
cască – bronz, armură - aur
cască – argint, armură - bronz
cască – argint, armură – argint

Indicați care este cea de-a opta combinație generată.

a. cască – argint, armură - aur
b. cască – aur, armură - argint
c. cască – argint, armură - platină
d. cască – aur, armură - bronz

5. Indicați valoarea afișată pe ecran după execuția programului următor.

```
void main(){
      int i = 0,c = 0;
      char s[20] = "ACESTA ESTE UN SIR";
      for (i = 0; i < 6; i++)
            c = c + 1;
      printf("%c", s[c + 1] + 1);
}
```

a. E
b. A
c. S
d. F

6. Indicați valoarea afișată pe ecran după execuția programului următor.

```
void main(){
      int ans = 0, i = 0, b = 1, j = 0;
      int N = 10050;
      while (N != 0)
      {
            b = 1;
            if (N % 10 == 0)
            {
                  for (j = 0; j < i; j++)
                        b = b * 10;
                  ans = ans + 1 * b;
            }
            else
            {
                  for (j = 0; j < i; j++)
                        b = b * 10;
                  ans = ans + (N % 10) * b;
            }
            N = N / 10;
            i++;
      }
      printf("%d\n", ans);
}
```

a. 11151
b. 10050
c. 01151
d. 11613

7. Câte componente conexe conține graful descris prin matricea de adiacență de mai jos?

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

a. 1
b. 3
c. 2
d. 4

8. Fie *v*, un vector de *n* elemente întregi, sortate crescător. Se dorește inserarea în vector a unui nou element, *x*, păstrând proprietatea ca vectorul să fie sortat crescător. Menționați cu ce trebuie înlocuită linia punctată pentru a îndeplini cerința.

```
int i;
for (i = n; i > 0; i--)
{
      ......................
            v[i] = v[i - 1];
      else
      {
            v[i] = x;
            break;
      }
}
if (v[0] > x)
      v[0] = x;
n++;
```

a. `if (v[i - 1] > x)`
b. `for(j=0;j<x;j++)`
c. `if (v[i] < x)`
d. `while(i>0)`

9. Precizați ce afișează programul următor.

```
int f(int x)
{
      if (x == 0)
            return 10;
      else
            if (x % 5 == 0)
                  return f(x / 10) * 10 + x;
            else
                  return f(x / 10) * (x % 10);
}
void main()
{
      printf("%d", f(2504));
}
```

a. 2500
b. 4400
c. 10000
d. 2000

SEPTEMBRIE 2024

1. Indicați valoarea afișată pe ecran după execuția secvenței de cod de mai jos:

```
void fun(char str1[], char str2[], int index){
    str2[index] = str1[index];
    if (index==10)
        return;
    fun(str1, str2, index + 1);
}
int main(){
    char str1[20]="ANA ARE MERE", str2[20]={0};
    fun(str1, str2, 0);
    printf("%s ", str2);
    return 0;
}
```

a. ANA ARE
b. nu se afișează nimic
c. ANA ARE MER
d. ANA ARE MERE

2. Care este suma gradelor tuturor nodurilor grafului descris prin matricea de adiacență de mai jos?

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

a. 8
b. 16
c. 15
d. 14

3. Funcția de mai jos trebuie să inverseze poziția elementelor vectorului *arr* (primul element devine ultimul, al doilea penultimul ș.a.m.d.).

```
void invers (int arr[], int n){
    int i, j;
    for (……; i < j; i++, j--) {
        int ele = arr[i];
        arr[i] = arr[j];
        arr[j] = ele;
    }
}
```

Cu ce trebuie înlocuită zona marcată prin ..... pentru ca funcția să îndeplinească în mod corect sarcina?

a. `i = 0, j = n`
b. `i = 0, j = n - 1`
c. `i = 1, j = n - 1`
d. `i = n - 1, j = 1`

4. Fie secvența de cod de mai jos. Menționați cu ce trebuie înlocuită linia punctată, astfel încât pe ecran să se afișeze interclasarea vectorilor *x* și *y*, adică *"1 2 3 4 5 6 7 8"*.

```
int x[10] = { 1,3,5,7,0,0,0,0,0,0 };
int y[10] = { 2,4,6,8,0,0,0,0,0,0 };
int z[10] = { 0,0,0,0,0,0,0,0,0,0 };
int i = 0, j = 0, k=0;
while (i < 4 && j < 4)
{
      if (x[i] < y[j])
            z[k++] = x[i++];
      else
            z[k++] = y[j++];
}
.....................
for (i = 0; i < 8; i++)
      printf("%d ", z[i]);
```

a. `z[k]=y[ j ];`
b. `z[k]=x[ i+1 ];`
c. `z[k]=x[ i ];`
d. `z[k]=y[ j+1 ];`

5. Se consideră secvența de cod de mai jos. Selectați afirmația corectă referitoare la vectorul *v* rezultat.

```
int v[100];
int i;
for (i = 0; i < 100; i++)
      v[i] = i;
for (i = 0; i < 100; i++)
      v[i] *= 2;
```

a. conține doar elemente mai mici decât 100
b. conține toate valorile de la 0 la 100
c. conține doar elemente pare
d. conține doar valori de 2

6. Menționați ce se afișează în urma execuției secvenței de cod de mai jos:

```
int i;
char str[100] = "Am zis:";
char s1[] = "Ha";
char s2[] = " ha";
strcat(str, s1);
for (i = 0; i < 2; i++)
     strcat(str, s2);
printf("%s", str);
```

a. Am zis:Ha ha!
b. Am zis:ha ha ha
c. Am zis:Ha ha
d. Am zis:Ha ha ha

7. Indicați ce se afișează pe ecran după execuția secvenței de cod de mai jos:

```
int f (int x, int y) {
      if (x == y) return x;

      if (x > y) return f(x - y, y);
      else return f(x, y - x);
}
void main() {
      int x = 25, y = 10;
      printf("%d\n", f(x, y));
}
```

a. cel mai mare divizor comun al lui x si y
b. maximul dintre x si y
c. minimul dintre x si y
d. cel mai mic multiplu comun al lui x si y

8. Precizaţi ce verifică funcţia de mai jos când este aplicată asupra unei matrice pătratice oarecare *A* de dimensiune *n* cu $n \leq 100$:

```
void check_matrix(int A[ ][100], int  n)
{
     int i, j, s = 0;
     for (i = 0; i < n; i++)
          for (j = 0; j < i; j++)
               if (A[i][j] == A[j][i])
                   s += 1;

     if (s  >=  n * (n - 1)/2)
          printf("verify: true\n");
     else
          printf("verify: false\n");
}
```

a. matricea conţine valorile şirului 1, 2, 3, 4, ..., (n – 1)
b. matricea este identică cu transpusa ei
c. matricea conţine cel puţin n * (n – 1)/2 elemente diferite de 0
d. matricea conţine cel puţin n * (n – 1)/2 elemente identice

9. Precizaţi ce afişează secvenţa următoare de cod:

```
int x = 2024, y = 3, z;
z = ((x-- / y) % ++y) * y;
printf("%d, %d , %d ", x, y, z);
```

a. 2023, 4, 8
b. 2023, 4, 7
c. 2024, 3, 7
d. 2023, 3, 8

SOLUȚII LA CHESTIONARELE PENTRU ADMITERE

Iulie 2021

1 a)	4 b)	7 c)
2 c)	5 d)	8 a)
3 e)	6 e)	9 d)

Iulie 2022

1 a)	4 e)	7 a)
2 d)	5 d)	8 c)
3 c)	6 c)	9 e)

Iulie 2023

1 c)	4 d)	7 b)
2 a)	5 a)	8 a)
3 b)	6 c)	9 d)

Septembrie 2023

1 b)	4 a)	7 c)
2 d)	5 d)	8 a)
3 c)	6 c)	9 b)

Iulie 2024

1 b)	4 d)	7 c)
2 b)	5 d)	8 a)
3 a)	6 a)	9 c)

Septembrie 2024

1 c)	4 a)	7 a)
2 b)	5 c)	8 b)
3 b)	6 d)	9 a)