

Subiecte la testul grilă de Informatică

1. Se dă o variabilă x care reține un număr natural nenul. Câte din expresiile de mai jos (scrise în C/C++) au valoarea 0 (false) pentru orice număr natural nenul x ?

- i) $(x/5 + x/6) \% 9$
- ii) $(x \% 100 + x \% 100 / 10) / 10$
- iii) $((10 - x \% 10) + (10 - x \% 100 / 10)) / 10$
- iv) $(x \% 5 + x \% 7) / 10$
- v) $(x \% 3 + x \% 6) / 8$
- vi) $(x \% 10 + x / 10) / 9$

- (a) 3 (b) 1 (c) 2 (d) 4

2. Fie structura:

```
struct INFO {  
    int val, a, b;  
};
```

și funcția:

```
void f3(INFO v)  
{  
    {  
        int aux;  
        cin >> v.val;  
        v.a = v.val % 5;  
        do {  
            v.b = 0;  
            for (aux = v.val; aux != 0; aux /= 10)  
            {  
                v.b = v.b + aux % 10;  
            }  
            v.val = v.b;  
        } while (v.b >= 10);  
    }  
}
```

apelată în secvența de cod notată **L1**:

```
INFO v = {0, 0, 0};  
f3(v);  
cout << v.val << " " << v.a << " " << v.b << endl;
```

Ce se afișează la execuția secvenței de cod **L1** dacă se citește de la tastatură valoarea 27011914?

- (a) 0 0 0
- (b) 7 4 7

- (c) Nu se afișează nimic. Se intră într-o buclă infinită.
(d) 27011914 4 7

3. Care este valoarea returnată de funcția fun apelată cu valoarea 4 pentru parametrul x ?

```
int fun(int x) {  
    int i = -10;  
    for (i = -3; i < 4 * x; i += x) {  
        if ((i < 10) || (i > 16)) {  
            } else {  
                break;  
            }  
        }  
    }  
    return i;  
}
```

- (a) 9 (b) 13 (c) 10 (d) 12

4. Fie următoarele declarații în cod C/C++:

```
typedef struct Student {  
    int nota;  
    char nume[30];  
} stud;
```

```
struct Grupa {  
    int nr_grupa;  
    stud studenti[12];  
};
```

```
struct Grupa grupa;
```

Alegeți care dintre următoarele instrucțiuni specifică faptul că al doilea student din listă primește nota 10.

- (a) grupa[1].studenti.nota = 10;
(b) grupa.studenti[1].nota = 10;
(c) grupa.studenti.nota = 10;
(d) grupa.studenti[2].nota = 10;

5. Ce se va afișa în urma executării secvenței de cod de mai jos, considerând că *i* este de tip întreg, iar *s* și *x* pot memora un șir format din maxim 25 de caractere.

```
strcpy(s, "testAdmitere2025");  
for(i = 1; i < strlen(s) / 2; i++){  
    strcpy(x, s);  
    x[i] = '\\0';  
    strcpy(s, s + i);  
    strcat(s, x);  
}  
cout << s;
```

- (a) 2025testAdmitere (b) itere2025testAdm (c) Admitere2025test (d) tere2025testAdmi

6. Știind că vectorul *v* are doar elemente numere naturale de la 0 la 9 și că elementele vectorului *x* au fost inițializate cu 0, la ce se poate folosi secvența următoare de instrucțiuni? Vectorul *v* are maxim 10 elemente, indexul primului element fiind 0.

```

for (i = 0; i < 10; i++)
{
    x[v[i]]++;
}

```

- (a) Se numără frecvența elementelor vectorului v.
- (b) Se incrementează valoarea elementelor vectorului v.
- (c) Se adună la valorile vectorului v elementele vectorului x.
- (d) Se inițializează toate elementele vectorului x cu 1, indiferent de valoarea elementelor vectorului v.

7. Un cămin de studenți are 4 etaje, iar pe fiecare etaj sunt 10 camere. În fiecare cameră stau maxim 4 studenți. Pentru a ține această evidență se folosește o matrice *mat*, cu indicele primului element 0, 0. Dacă se dorește afișarea numărului total de studenți de pe fiecare etaj, care dintre secvențele de cod de mai jos poate fi folosită?

(a)

```

for (i = 0; i < 4; i++)
{
    rez = 0;
    for (j = 0; j < 10; j++)
        rez = rez + mat[i][j];
    cout << "Pe etajul " << i << " sunt " << rez << " studenti" << endl;
}

```

(b)

```

for (i = 0; i < 4; i++)
{
    for (j = 0; j < 10; j++)
    {
        rez = 0;
        rez = rez + mat[i][j];
    }
    cout << "Pe etajul " << i << " sunt " << rez << " studenti" << endl;
}

```

(c)

```

for (j = 0; j < 4; j++)
{
    rez = 0;
    for (i = 0; i < 10; i++)
    {
        rez = rez + mat[i][j];
        cout << "Pe etajul " << i << " sunt " << rez << " studenti" << endl;
    }
}

```

(d)

```

for (j = 0; j < 4; j++)
{
    rez = 0;
    for (i = 0; i < 10; i++)
    {
        rez = rez + mat[i][j];
    }
    cout << "Pe etajul " << i << " sunt " << rez << " studenti" << endl;
}

```

8. Câte interschimbări ale valorilor elementelor matricei `mat` au loc la apelul funcției `transform(mat)`?

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

void transform(int mat[4][4]) {
    for (int i = 0; i < 4; i++) {
        for (int j = 3; j >= i+1; j--) {
            int temp = mat[i][j];
            mat[i][j] = mat[j][i];
            mat[j][i] = temp;
        }
    }
}
```

(a) 12 (b) 8 (c) 9 (d) 6

9. Care sunt elementele de pe linia cu indexul 2 a matricei `mat` după apelul funcției `transform(mat)`?

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

void transform(int mat[4][4]) {
    int temp[4][4];
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            temp[j][3 - i] = mat[i][j];
        }
    }

    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            mat[i][j] = temp[i][j];
        }
    }
}
```

(a) {9,7,1,6} (b) {2,3,8,6} (c) {6,8,3,2} (d) {6,1,7,9}

10. Pentru care dintre următoarele tablouri funcția `process` returnează valoarea cea mai mare? Apelul funcției se face considerând al doilea parametru ca fiind dimensiunea tabloului în fiecare caz.

```
int process(int v[], int n) {
    int m = 1, c = 1;
    for (int i = 1; i < n; i++) {
        if (v[i] > v[i - 1]) {
```

```

        c++;
        if (c > m) {
            m = c;
        }
    } else {
        c = 1;
    }
}
return m;
}

```

- (a) {1, 2, 2, 3, 5, 7, 1, 2, 3, 4}
- (b) {1, 1, 7, 3, 2, 4, 6, 8, 2, 3}
- (c) {3, 2, 4, 1, 2, 5, 6, 4, 8, 9, 3, 5, 1}
- (d) {8, 0, 1, 3, 4, 9, 1, 2, 2, 3, 4}

11. Fie structura:

```

struct INFO {
    int val, w1, w2;
};

```

și funcțiile:

```

void f1(INFO v[], int lim)
{
    int i;
    for(i=0; i<lim; ++i)
    {
        int aux;
        cin >> v[i].val;
        v[i].w1 = i;
        v[i].w2 = 0;
        for(aux = v[i].val; aux !=0; aux /= 10)
            v[i].w2 = v[i].w2 + aux%10;
    }
}

```

```

void f2(INFO v[], int lim)
{
    int i, flag, k = lim - 1;
    INFO aux;
    do {
        flag = 0;
        for(i = 0; i < k; ++i) {
            if(v[i].w2 < v[i+1].w2) {
                flag = 1;
                aux = v[i];
                v[i] = v[i+1];
                v[i+1] = aux;
            }
        }
        k--;
    } while(flag == 1);
}

```

apelate în secvența de cod:

```
INFO v[20];  
int n = 11;  
f1(v, n);  
f2(v, n);
```

La execuția secvenței de cod se citesc, de la tastatură, următoarele valori:
85885 62036 25234 52327 34763 45088 36512 99753 30184 83298 71849

Care este valoarea membrului $w1$, după rularea secvenței de cod, a ultimului element al vectorului de structuri?

(a) 8 (b) 10 (c) 7 (d) 0

12. Numărul de submulțimi al unei mulțimi finite $M = \{a_1, a_2, \dots, a_n\}$ este:

(a) 2^n (b) n^2 (c) n (d) 1

13. Vrăjitorul din Oz vrea să creeze o vraja care să o trimită pe Dorothy acasă. Pentru aceasta roagă pe un candidat de la Facultatea de Automatică și Calculatoare din Iași să îi scrie un program care să îl ajute să amestece aceste elemente într-o anumită ordine. Soluția realizată este următoarea:

```
#include <iostream>  
using namespace std;  
int x[10], n=7;  
char magic[][7] = { "Apa", "Aer", "Foc", "Vant", "Pamant", "Metal", "Lemn" };  
int Validare(int k){  
    for(int i=0; i<k; i++)  
        if(x[i]==x[k]) return 0;  
    return 1;  
}  
void Afisare(int k){  
    for(int i=0; i<k; i++)  
        cout<<magic[x[i]]<<" ";  
    cout<<endl;  
}  
void Back(int k){  
    for(int i = 0 ; i < n ; ++i) {  
        x[k]=i;  
        if( Validare(k) )  
            if(k==n-1) Afisare(k);  
            else Back(k+1);  
    }  
}  
int main() {  
    Back(0);  
    return 0;  
}
```

Să se indice câte dintre elemente vor fi afișate în fiecare soluție.

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

14. Răzvan este elev în clasa a III-a și în fiecare dimineață trebuie să se pregătească pentru școală. El trebuie să facă o serie de acțiuni: (1) să închidă ghiozdanul, (2) să pună penarul în ghiozdan, (3) să pună tema pentru acasă în ghiozdan, (4) să se spele pe dinți, (5) să se îmbrace cu hainele de școală, (6) să umple sticla cu apă, (7) să mănânce, (8) să se spele pe mâini, (9) să pună sticla cu apă în ghiozdan.

Penarul, tema și sticla umplută cu apă trebuie puse în ghiozdan și abia apoi poate fi închis ghiozdanul. Sticla cu apă trebuie umplută înainte de a fi pusă în ghiozdan. Înainte de a mânca, trebuie să se spele pe mâini, iar după masă trebuie să se spele pe dinți.

Știind că cele nouă acțiuni pot fi făcute de Răzvan doar secvențial, în ce ordine ar trebui efectuate astfel încât să respecte restricțiile menționate. De-un caz că ar putea să aleagă la un moment dat din mai multe acțiuni, Răzvan alege să îl facă întâi pe ce cu numărul de ordine mai mic.

- (a) 2, 3, 1, 5, 6, 8, 7, 9, 4 (b) 2, 3, 5, 4, 9, 1, 8, 7, 6 (c) 2, 3, 1, 5, 6, 8, 7, 9, 4 (d) 2, 3, 5, 6, 8, 7, 4, 9, 1

15. Care este complexitatea de timp a funcției de mai jos?

```
long compute(int n) {
    int i, j;
    long res;
    for (i=n; i>0; i/=2) {
        for (j=i; j>0; j/=2) {
            res += i*j;
        }
    }
    return res;
}
```

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

16. Un triunghi dreptunghic cu perimetrul p și laturile $\{a, b, c\}$ are proprietatea $c^2 = a^2 + b^2$. Pentru un anumit perimetru p pot fi mai multe triunghiuri dreptunghice. De exemplu, dacă $p=120$ triunghiurile dreptunghice pot avea laturile $\{20, 48, 52\}$, $\{24, 45, 51\}$, $\{30, 40, 50\}$. Funcția de mai jos calculează câte triunghiuri dreptunghice au un perimetru dat și afișează laturile acestora. Alegeti instrucțiunile lipsă marcate cu X, Y și Z pentru ca funcția să fie corectă.

```
int nb_of_triangles(int p){
    int a, b, c;
    int nt = 0;
    for (a = 1; X; a++)
        for (Y; b++) {
            Z;
            if (c * c == a * a + b * b) {
                nt++;
                cout<<a<<" "<<b<<" "<<c<<endl;
            }
        }
    return nt;
}
```

- (a) X: $a \leq 1 + p / 3$
 Y: $b = 1 + a$; $b \leq 1 + (p - a) / 2$
 Z: $c = p - a - b$
- (b) X: $a \leq p/3$
 Y: $b = 1 + a$; $b \leq 1 + (p - a)$
 Z: $c = p - (a + b)/2$
- (c) X: $a < 1 + p / 3$
 Y: $b = a$; $b < 1 + (p - a)$
 Z: $c = p - (a + b)/2$
- (d) X: $a \leq p / 3$
 Y: $b = a$; $b \leq (p - a) / 2$
 Z: $c = p/2 - a - b$

17. Fie secvența de cod de mai jos:

```
int i, a, nr=0, n=20;
for (i=1; i<=n; ++i){
    a=i;
    while (a!=0){
        nr=nr+a%2;
        a=a/2;
    }
    cout<<nr<<" ";
}
```

Care sunt a treia, a șasea și ultimile 2 valori afișate?

(a) 4,9,40,42 (b) 2,2,3,2 (c) 3,6,19,20 (d) 4,9,3,22

18. Fie mulțimea $M = \{1, 2, 3, 6, 7\}$. Un algoritm generează toate numerele naturale de k cifre, folosind cifre distincte din mulțimea M , care nu au alăturate cifre de aceeași paritate. Pentru $k = 4$, primele patru soluții generate în această ordine sunt: 1236, 1276, 1632, 1672, câte soluții vor fi generate de algoritm?

(a) 12 (b) 24 (c) 16 (d) 20

19. Un anagajat are de efectuat $N = 7$ task-uri, fiecare caracterizat de un timp necesar pentru finalizare și un deadline (termen limită). Fiecare task durează exact t unități de timp pentru a fi finalizat și trebuie terminat înainte de deadline-ul său d . La un moment dat se poate executa doar un singur task. Trebuie rezolvate cât mai multe task-uri în timp util. Dacă taskurile sunt definite astfel $(t, d) : (3, 6), (2, 4), (1, 3), (6, 12), (2, 6), (3, 7), (1, 2)$, se cere care este timpul total, necesar pentru a finaliza un număr maxim de taskuri în condițiile date?

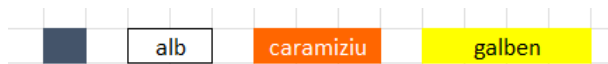
(a) 12 (b) 9 (c) 10 (d) 18

20. La o tombolă participă n persoane, fiecare participant având un număr de ordine (de la 1 la n). Câștigătorii sunt selectați după următoarea regulă: se alege fiecare al k -lea ($0 < k < n$) participant din cei n , numărând din k în k . Când numărătoarea ajunge la ultimul participant, se continuă cu primul, numărând mai departe toate persoanele, fie că au fost sau nu selectate. Numărătoarea se oprește atunci când ar trebui selectat un participant deja declarat câștigător. Dacă se cunosc valorile n și k , care este formula cu care se determină câți participanți nu sunt selectați drept câștigători?

Funcția $cmmmc$ calculează cel mai mic multiplu comun, iar funcția $cmmdc$ calculează cel mai mare divizor comun.

- (a) $n - cmmmc(n, k - 1)/k$
- (b) $n - n/cmmdc(n, k)$
- (c) $n - k/cmmdc(n, k)$
- (d) $n - cmmmc(n, k)$

21. Intr-o grădină botanică, un peisagist trebuie să construiască o alee lată de 50 cm din dale pătrate sau dreptunghiulare de înălțime 10 cm dar de culori și lățimi diferite: gri – 10 cm, albe – 20 cm, cărămizii – 30 cm, galbene – 40 cm.



Peisagistul hotărăște să așeze dalele amestecat, în rânduri de dimensiune 10 x 50 cm. Câte rânduri diferite poate obține și câte dale gri va utiliza pentru acest proiect?

(a) 12, 26 (b) 18, 32 (c) 13, 28 (d) 15, 28

22. Joy și-a planificat o excursie de o zi în orașul Dezorientat, unde va ajunge dimineața cu trenul. Joy își dorește să viziteze toate obiectivele turistice, așadar se folosește de harta orașului. Pe aceasta sunt marcate cele mai importante atracții. Pentru a fi eficientă, Joy dorește să treacă pe la fiecare obiectiv turistic o singură dată și la final să se întoarcă la gară pentru a lua trenul către o altă destinație. Care din condițiile de mai jos trebuie îndeplinită pentru ca Joy să își poată îndeplini obiectivul propus?
- (a) Joy poate să viziteze toate obiectivele turistice o singură dată și să se întoarcă la gară doar dacă harta orașului este modelată ca un graf eulerian.
 - (b) Joy poate să viziteze toate obiectivele turistice o singură dată și să se întoarcă la gară doar dacă harta orașului este modelată ca un graf hamiltonian.
 - (c) Joy poate să viziteze toate obiectivele turistice o singură dată și să se întoarcă la gară doar dacă există un ciclu în graful aferent hărții.
 - (d) Joy poate să viziteze toate obiectivele turistice o singură dată și să se întoarcă la gară doar dacă harta orașului este modelată ca un graf complet.

23. Fie G un graf neorientat cu m noduri, care este complet. Presupunem că avem un număr de n astfel de grafuri, notate cu $G_1, G_2, \dots, G_n$. Aceste grafuri sunt conectate într-o structură S de tip lanț de grafuri: $G_1 - G_2 - \dots - G_n$. Conectarea a două grafuri adiacente din structura S se face printr-o singură muchie. Presupunem că pentru toate grafurile G_k din structura S , nodurile prin care se face conectarea cu grafurile adiacente sunt diferite. Care este diametrul structurii S ?

Definiții:

$G = (V, E)$ - graf neorientat cu setul de noduri V și setul de muchii E .

$d(u, v)$ - distanța minimă dintre nodurile u și v , egală cu numărul de muchii al celui mai scurt drum dintre u și v .

$\epsilon(u) = \max_{v \in V} d(u, v)$ - excentricitatea nodului u .

$radius(G) = \min_{u \in V} \epsilon(u)$ - raza grafului G .

$diameter(G) = \max_{u \in V} \epsilon(u)$ - diametrul grafului G .

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

24. Care este valoarea returnată după apelul funcției *process*($v, 6$)?

```
int v[] = {2, 1, 7, 2, 9, 6};
```

```
int process(int v[], int n) {
    int x = 0;
    if (n == 0) return 0;
    if (v[n-1] % 2 == 0) {
        x = v[n-1];
    }
    return x + process(v, n-1);
}
```

- (a) 27 (b) 18 (c) 0 (d) 10

25. Ce verifică funcția **verif** definită mai jos, presupunând că apelul inițial este *verif*($n, 1, 1$).

```
int verif(int n, int x, int k)
{
    if (k == n)
        return 1;
    if (k > n)
        return 0;

    return verif(n, x+1, k*x);
}
```

- (a) Funcția verifică dacă n este un număr prim.
- (b) Funcția verifică dacă n poate fi exprimat ca produsul de numere consecutive începând cu x .
- (c) Funcția verifică dacă n poate fi exprimat ca sumă de numere consecutive începând cu x .
- (d) Funcția verifică dacă n este un pătrat perfect.