

Subiecte la testul grilă de Informatică

1. Se consideră un vector *vec* de numere naturale cu *n* elemente. Care din algoritmii următori descriu pașii necesari calculării mediei aritmetice a elementelor vectorului?

(a)
`s = 0;
 for (i = 0; i < n; i++)
 {
 s = s + vec[i];
 }
 s = s/n;`

(b)
`s = 0;
 for (i = 0; i < n; i++)
 {
 s = s + vec[i];
 s = s/n;
 }`

(c)
`s = 0;
 for (i = 0; i < n; i++)
 {
 s = s + i;
 s = s/n;
 }`

(d)
`s = 0;
 for (i = 0; i < n; i++)
 {
 s = s + i;
 }
 s = s/n;`

2. Ce se va afișa după rularea secvenței de cod de mai jos, dacă de la tastatură se vor citi valorile 10 și 15 în această ordine?

```
int main()
{
    int x,y;
    cin>>x>>y;
    while (x == y)
    {
        if (x>y)
            x = x - y;
        else
            y = y - x;
    }
    cout<<x;
    return 0;
}
```

- (a) 5 (b) 10 (c) 15 (d) 0

3. Se consideră un sir de numere întregi care se termină cu valoarea zero, de forma a1, ..., an, 0. Se cere determinarea produsului numerelor introduse. Pentru secvența de cod de mai jos, să se analizeze posibilitatea ca instrucțiunea while să fie înlocuită cu instrucțiunea do ... while în cazul introducerii de la tastatură a sirului de valori 4 2 1 3 0.

```
int a = 1, p = 1;
while (a != 0 )
```

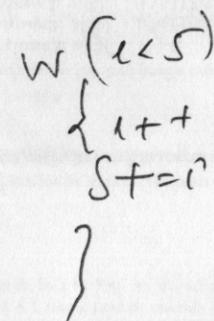
```
{
    cin >> a;
    p = p * a;
}
cout << "p = " << p; //printf("p = %d", p);
```

Care dintre următoarele afirmații este adevărată?

- (a) instrucțiunea while poate fi înlocuită cu do ... while, iar p = 24
 (b) Instrucțiunea while nu poate fi înlocuită cu do ... while
 (c) instrucțiunea while nu poate fi înlocuită cu do ... while, iar p = 24
 (d) instrucțiunea while poate fi înlocuită cu do ... while, iar p = 0

4. Ce se va afișa la rularea secvenței următoare?

```
int i, s = 5;
for (i = 0, s = 0; i < 5; i = i + 2)
{
    while (i++ < 5)
        s = s + i;
    cout << "s = " << s << endl;
}
```



- (a) s = 10
 (b) s = 15
 s = 30
 s = 45
 (c) s = 15
 (d) s = 20

5. Ce se va afișa la rularea secvenței următoare?

```
int i, j;
char c = 'A';
for(i = 3; i > 0; i--)
{
    for(j = i; j >= 0; j--)
        cout << (char)(c + j);
    cout << " ";
}
```

- (a) DCB CB B (b) DCBA CBA BA (c) CBA BA A (d) DCBA CBA BA A

6. Ce se afișază la rularea următoarei secvențe de cod? Se consideră că indexul primului element al vectorului este 0 (zero).

```
#include <iostream>
using namespace std;
int main(void)
{
    int a[5] = {1, 2, 3, 4, 5};
    int i, j, m;
    i = ++a[2];
    j = a[1]++;
}
```

```

m = a[i++];
cout << i << " , " << j << " , ";
cout << m << endl;
return 0;
}

```

- (a) 5, 2, 5
 (b) 4, 2, 5
 (c) 4, 2, 4
 (d) 5, 3, 5

7. Pentru a calcula în mod eficient media aritmetică a elementelor diagonalei principale a unui tablou bidimensional pătratic de dimensiune n cu componente numere naturale este necesar și suficient să se execute:

- (a) O singură instrucțiune de atribuire
 (b) O singură parcurgere a diagonalei principale și o atribuire
 (c) O singură parcurgere a diagonalei principale și n atribuirii
 (d) O singură parcurgere a tabloului și $n + 1$ atribuirii

8. Se consideră un vector cu $n = 8$ elemente de tip întreg, declarat și inițializat astfel (indexul primului element este 0):

```
int vec[] = {1, 4, 5, 4, 2, 4, 6, 7};
```

Cum va arata vectorul după rularea următorului cod?

```

for (i = 0; i < n; i++)
  if (i % 3 == 0)
    vec[i] = 0;

```

- (a) 0 4 5 0 2 4 0 7
 (b) 1 4 5 4 2 4 0 7
 (c) 1 0 5 0 0 0 0 7
 (d) 0 4 0 4 2 4 6 0

9. Fie o matrice cu 5 linii și 5 coloane notată în secvență de mai jos *matrice*. Elementul de pe prima linie și prima coloană aflat pe poziția 0, 0:

```

1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5

```

Ce va afișa următoarea funcție dacă va fi apelată cu valoarea 2?

```

void functie(int val)
{
  int i, j, s = 0;
  for (i = 0; i < 5; i++)
    s = s + matrice[i][val];
  for (i = 0; i < 5; i++)
    s = s + matrice[val][i];
  cout << "s = " << s;
}

```

- (a) 25 (b) 30 (c) 35 (d) 40

10. Se consideră următoarea funcție reprezentată în pseudocod (s-a notat cu a un tablou bidimensional cu n linii și m coloane):

```

funcția f(a, n, k)
  s ← 0
  pentru i ← 1 la n execută
    s ← s + a(i, k)
  returnează s

```

și procedura *Algoritm* (scrisă de asemenea în pseudocod)

```

procedura Algoritm
  k ← 1
  V ← f(a, n, 1)
  pentru i ← 2 la m execută
    val ← f(a, n, i)
    dacă val > V atunci
      V ← val
      k ← i
  scrie k

```

Ce se afișează la rularea programului care implementează pseudocodul dat?

- (a) Indicele coloanei de sumă maximă
 (b) Indicele coloanei de sumă minimă
 (c) Indicele liniei de sumă maximă
 (d) Indicele liniei de sumă minimă

11. Fie secvența de program:

```

for (i=1;i<=n;i++)
  instructiune1
  {
    instructiune2
    a[j][i]=a[i][j];
  }

```

Care sunt instrucțiunile ce lipsesc pentru ca după execuție, în cazul în care $n = 4$, matricea să aibă valorile:
 1 2 3 0
 2 3 0 1
 3 0 1 2
 0 1 2 3

- (a) **instructiune1:** for (j=1; j<=n; j++) **instructiune2:** a[i][j] = i+j-1;
 (b) **instructiune1:** for (j=i; j<=n; j++) **instructiune2:** a[i][j] = (i*n+j)%n;
 (c) **instructiune1:** for (j=i; j<=n; j++) **instructiune2:** a[i][j] = (i+j-1)%n;
 (d) **instructiune1:** for (j=i; j<=n; j++) **instructiune2:** a[i][j] = (i*n+j-1)%n;

12. Se consideră algoritmul de sortare crescătoare prin metoda bulelor aplicat la secvența de numere 2,4,1,7,3,5. Care este secvența obținută după prima parcurgere?

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

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

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

- (a) $O(n)$ (b) $O(\log_2(n))$ (c) $O(n^2)$ (d) $O(n \cdot \log_2(n))$

14. Fie 6 siruri de caractere (*prune*, *portocale*, *mere*, *ananas*, *smochine*, *clementine*) și un program care concatenează sirurile două câte două, astfel încât, în final, să rezulte un sir format din toate cuvintele, dar nu neapărat în ordinea inițială. Programul concatenează cuvintele astfel încât numărul de accesări ale literelor să fie minim.

De exemplu, pentru concatenarea cuvintelor *prune* și *portocale* sunt necesare 14 accesări (5 pentru accesarea literelor cuvântului *prune* și 9 pentru accesarea cuvântului *portocale*). Dacă acest rezultat (*pruneportocale*) este concatenat cu *mere*, atunci va rezulta sirul *pruneportocalemere* în 18 accesări.

Care este numărul minim de accesări pentru a concatena două câte două cele 6 cuvinte astfel încât să rezulte un sir format din toate cuvintele (Atenție! cuvintele nu trebuie concatenate neapărat în ordinea în care au fost date).

- (a) 123 (b) 42 (c) 107 (d) 130

15. Ce valoare va avea variabila *count* în urma execuției funcției *f*?

```
int n = 6;
int a[23];
int count = 0;

void f()
{
    int i;
    if(a[0])
        count++;
    for(i=a[a[0]]+1; i<=n; i++)
    {
        a[i+a[0]] = i; f(); a[a[0]]--;
    }
}
```

- (a) 64 (b) 63 (c) 32 (d) 31

16. Citind despre spioni și despre diverse modalități de a codifica mesajele, Mickey și Minnie, se hotărăsc să își codifice mesajele. Pentru ca mesajul lor să nu poată fi înțeles de altcineva, ei procedează astfel: aleg împreună un cuvânt numit "cheie", format numai din litere mari, distincte. Apoi, împart mesajul pe care doresc să-l transmită între ei în secvențe de caractere alăturate de lungime egală cu numărul de litere ale cuvântului cheie. Scriu pe foaie cuvântul cheie ales și sub acesta se scriu secvențele determinate anterior, în ordinea obținerii lor. Desigur, există posibilitatea ca ultima grupă să fie incompletă. Mesajul codificat se obține astfel: se parcurge tabelul obținut anterior, pe coloane, de sus în jos, ordinea de parcurgere a coloanelor este ordinea alfabetică a literelor din cuvântul cheie. Considerând cuvântul cheie **SUBIECT**, care este decodificarea mesajului **MLREITT AIAMAEFCRNID OA**:

- (a) INFORMATICA LA ADMITERE
(b) LA ADMITERE INFORMATICA
(c) ADMITERE LA INFORMATICA
(d) ADMITE REAL INFORMATICA

17. Se consideră un graf neorientat. Care dintre următoarele propoziții este adevărată?

P: Numărul de vârfuri de grad impar este par

Q: Suma gradelor tuturor vârfurilor este pară

- (a) Numai P (b) Numai Q (c) Să P și Q (d) Niște P și Q

18. Fie G un graf bipartit neorientat, cu cele două seturi de noduri conținând m , respectiv n noduri. Dacă G este bipartit complet, căte muchii conține?

Un graf bipartit neorientat complet este un graf în care fiecare pereche de noduri din cele două seturi (un nod din primul set, celălalt nod din al doilea set) este conectată printr-o muchie.

- (a) m^n (b) $m+n$ (c) $m \cdot n$ (d) C_m^n

19. Fie un arbore binar în care fiecare nod poate avea maxim doi fiți și, pentru fiecare nod, subarborele stâng conține valori mai mici decât cea a nodului, iar cel drept conține valori mai mari decât cea a nodului. În acest arbore se introduc N numere (N este divizibil la 4). Primele 25% dintre numerele introduse sunt în ordine crescătoare, iar următoarele 75% sunt numere mai mici decât primele și sunt introduse în ordine descrescătoare.

Care este înălțimea arborelui rezultat?

Înălțimea unui arbore este numărul de legături (arce) de pe cea mai lungă cale de la rădăcină la o frunză.

- (a) $3 * N/4$ (b) $3 * N/4 - 1$ (c) $\log_2 N$ (d) $3 * N/4 + 1$

20. Fie G un graf turneu. Care este numărul total de posibilități în care pot fi aranjate muchiile acestui graf?

Un graf turneu este un graf orientat în care fiecare pereche de noduri distincte este conectată printr-o singură muchie orientată.

- (a) $n!$ (b) $2^{n*(n-1)/2}$ (c) n^2 (d) C_n^2

21. Fie un graf neorientat cu 100 de vârfuri numerotate de la 1 la 100. Se știe că de la fiecare vârfuri k există muchie la vârfurile $[k/2]$ (partea întreagă din $k/2$), $2*k$ și $2*k+1$ (doar pentru valorile vârfurilor de la 1 la 100). De exemplu, de la vârful 7 există muchie la vârfurile 3, 14 și 15; în schimb, de la vârful 50 există muchie doar către vârfurile 25 și 100. Care este diametrul grafului? Diametrul unui graf este definit ca maximul distanței minime între două vârfuri. Distanța minimă între două vârfuri reprezintă numărul minim de muchii necesare pentru a ajunge de la un vârf la celălalt.

- (a) 10 (b) 12 (c) 100 (d) 6

22. Fie următorul pseudocod:

```
funcția f(x, y)
| dacă x = y atunci
| | returnează x + y
| altfel
| | returnează 1 + f(x + 1, y - 1)
```

Care este rezultatul apelului $f(1000, 1500)$?

- (a) 3500 (b) 2500 (c) 3000 (d) 2750

23. Se consideră următoarea funcție recursivă:

```
int fr (unsigned long int n)
{
    long int nl = n/10;
    if(nl == 0)
        return 1;
```

```

else
    return 1+f(n1);
}

```

Care dintre următoarele funcții nerecursive este echivalentă cu funcția *fr* (o funcție *f* este echivalentă cu o altă funcție *g* dacă pentru aceleși date de intrare se obțin aceleși rezultate)?

(a)

```

int f(unsigned long int n)
{
    int j = 0;
    do {
        j++;
        n /= 10;
    } while (n != 0);
    return j;
}

```

(b)

```

int f(unsigned long int n)
{
    int j = 0;
    while(n%10 == 0) {
        j++;
        n /= 10;
    }
    return j;
}

```

(c)

```

int f(unsigned long int n)
{
    int j = 0;
    while(n /10 != 0)
    {
        j++;
        n /= 10;
    }
    return j;
}

```

(d)

```

int f(unsigned long int n)
{
    int j = 0;
    do {
        j++;
        n /= 10;
    } while (n == 0);
    return j;
}

```

Q coada de la 0 la n-2

24. Se consideră o coadă circulară de capacitate $(n - 1)$ implementată static cu un vector de n elemente. Operațiile de inserare și stergere se realizează utilizând idici PRIM (indică poziția valorii ce poate fi ștersă) și ULTIM (poziția liberă unde se poate insera o valoare). O poziție va rămâne goală când coada este plină și dacă unul din indici ajunge pe ultima poziție din vector poate trece pe poziția 0 dacă aceasta este liberă. Inițial, PRIM = ULTIM = 0.

Condițiile necesare pentru a detecta coada plină și coada goală sunt:

- (a) Plina : $(ULTIM + 1) \bmod n = PRIM$, Goală : $ULTIM = PRIM$
- (b) Plina : $(ULTIM + 1) \bmod n = PRIM$, Goală : $(PRIM + 1) \bmod n = ULTIM$
- (c) Plina : $ULTIM = PRIM$, Goală : $(ULTIM + 1) \bmod n = PRIM$
- (d) Plina : $(PRIM + 1) \bmod n = ULTIM$, Goală : $ULTIM = PRIM$

25. Alecu se duce la bancă și depune s lei într-un cont special. Dobânda este de 0,5% pe lună și se depune în cont la sfârșitul fiecărei luni. Știind că funcția *main* este de forma:

```

int main(void)
{
    int n = 12;
    double s = 10000;
    s = f(s, n);
}

```

```

cout << s << endl;
return 0;
}

```

care dintre următoarele funcții îi permite lui Alecu să afle care va fi suma pe care poate să o retragă de la bancă după n luni?

I.

```

double f(double s, int n)
{
    if(n == 0)
    {
        return s;
    }
    else
    {
        return (1 + 0.5/100/12)
            * f(s, --n);
    }
}

```

II.

```

double f(double s, int n)
{
    if(n == 0)
    {
        return s;
    }
    else
    {
        return (1 + 0.5/100/12)
            * f(s, n--);
    }
}

```

III.

```

void f(double s, int n)
{
    double s1 = s;
    int i = 0;

    while (i <= n)
    {
        s1 *= (1 + 0.5/100/12);
        ++i;
    }
}

```

IV.

```

double f(double s, int n)
{
    double s1 = s;
    int i = 0;

    while (i < n)
    {
        s1 *= (1 + 0.5/100/12);
        i++;
    }
    return s1;
}

```

- (a) I și IV
- (b) I și II
- (c) III și IV
- (d) II și III