

```
int f1(int a, int b){
    if(b == 0)
        return 1;
    if(b % 2 == 0)
        return f1(a, b/2) * f1(a, b/2);
    return f1(a, b/2) * f1(a, b/2) * a;
}
```

```
int f2(int a
       if(b == 0)
           return 1;
       int put = f2(a, b/2);
       if(b % 2 == 0)
           return put * put;
       return put * put * a;
}
```



1. Dandu-se cele 2 sechete de cod de deasupra, alegeti raspunsurile corecte:

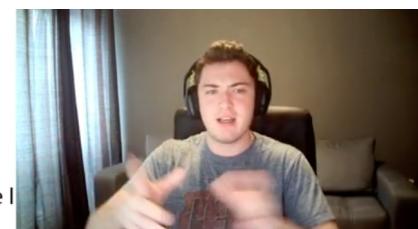
- A. Subprogramele f1 si f2 au aceiasi complexitate timp.
- B. Subprogramul f1 se incadreaza in clasa de complexitate $O(b)$.
- C. Subprogramul f2 se incadreaza in clasa de complexitate $O(\log_2(b))$.
- D. Cele 2 subprograme nu calculeaza acelasi lucru.

```
void scrie(int n){
    if(n == 0)
        return ;
    scrie(n-1);
    cout << n << ' ';
    scrie(n-1);
}
```



2. Selectati raspunsurile corecte:

- A. Pentru $n = 3$, se afiseaza 1 2 1 3 1 2 1
- B. Pentru $n = 0$, se afiseaza 0
- C. Pentru $n = 2$, se afiseaza 2 1 1 2
- D. Oricare ar fi n , se afiseaza 1 2 1 n 1 2 1



3. Se considera un tip de date intreg care se scrie pe x biti. Acesta poate fi:

- A. $[-2^x, -2^{(x-1)}-1]$
- B. $[-2^{(x-1)}+1, 2^{(x-1)}-1]$
- C. $[0, 2^{(x-1)}-1]$
- D. Toate raspunsurile sunt corecte

4. Gigel este prezent la un meci de fotbal. Din motive doar de el stiute, a intarziat si acum isi doreste sa afle in cate moduri s-ar fi putut ajunge la scorul curent. (prin numarul de moduri a se intlege numarul de succesiuni diferite de goluri marcate de cele 2 echipe pentru a se ajunge la scorul respectiv.)

Ex. $2 - 2$ s-ar fi putut obtine prin 6 succesiuni diferite.

Precizati care dintre urmatoarele variante de raspuns sunt corecte:

- A. Scorul $5 - 5$ s-ar fi putut obtine prin 250 de moduri
- B. Scorul $3 - 4$ s-ar fi putut obtine prin 35 de moduri
- C. Scorul $4 - 5$ s-ar fi putut obtine prin 126 de moduri
- D. Scorul $4 - 4$ s-ar fi putut obtine prin 68 de moduri



5. Precizati care dintre urmatoarele sechete de cod calculeaza corect numarul de cifre ale unui numar natural transmis ca parametru:

```
A. int nr_cifre(int n){
    if(n == 0)
        return 0;
    return 1 + nr_cifre(n / 10);
}
```

```
B. int nr_cifre(int n){
    int cnt = 0;
    do{
        cnt++;
        n /= 10;
    }while(n != 0);
    return cnt;
}
```

```
C. int nr_cifre(int n){
    int cnt = 0;
    while(n){
        cnt++;
        n /= 10;
    }
    return cnt;
}
```

```
D. int nr_cifre(int n){
    if(n < 10)
        return 1;
    return 1 + nr_cifre(n / 10);
}
```



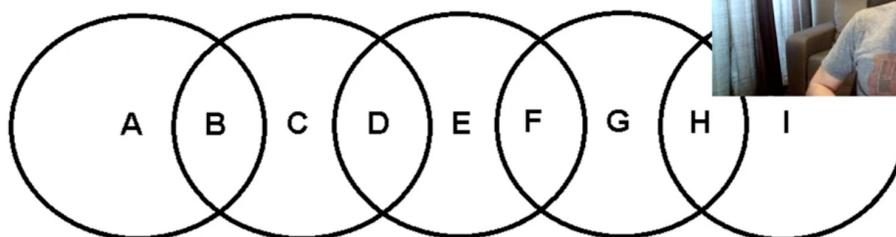
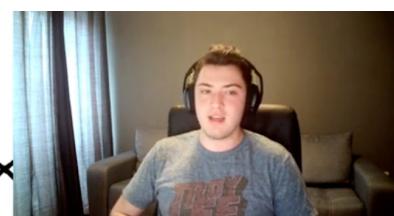
6. Precizati care dintre urmatorii subalgoritmi calculeaza corect C_n^k (limitele tipurilor de date nu se iau in calcul):

```
A. int Comb[1001][1001];
int C(int n, int k){
    if(n == 1 || n == k || k == 0)
        return 1;
    if(Comb[n-1][k-1] == 0)
        Comb[n-1][k-1] = C(n-1, k-1);
    if(Comb[n-1][k] == 0)
        Comb[n-1][k] = C(n-1, k);
    return Comb[n-1][k-1] + Comb[n-1][k];
}
```

```
B. int C(int n, int k){
    int prod = 1;
    for(int i = n - k + 1; i <= n; ++i)
        prod *= i;
    for(int i = 1; i <= k; ++i)
        prod /= i;
    return prod;
}
```

```
C. int C(int n, int k){
    if(n == 0 || k == 0)
        return 1;
    return C(n-1, k) + C(n-1, k-1);
}
```

```
D. int C(int n, int k){
    int a, b = 1;
    for(int i = 1; i <= k; ++i)
        b *= i;
    a = b;
    for(int i = k + 1; i <= n; ++i)
        a *= i;
    return a / b;
}
```



7. In desenul alaturat, trebuie sa alegeti valori distincte din multimea $\{1,2,3,4,5,6,7,8,9\}$ pentru valorile $\{A,B,C,D,E,F,G,H,I\}$ in asa fel incat suma elementelor din fiecare cerc sa fie aceiasi. Precizati care dintre urmatoarele afirmatii sunt corecte:

- A. Cu suma elementelor dintr-un cerc 13, exista 4 solutii.
- B. Cu sume elementelor dintr-un cerc 12, exista 2 solutii.
- C. Există soluții pentru sumele în cerc 11, 13, 12 și 14.
- D. În total există 10 solutii.



8. Se considera un sir v cu n ($1 \leq n \leq 1000$) elemente numere naturale. Pentru a rezolva problema se doreste identificarea urmatorului element mai mare decat el, existent in sir. Precizati care informatii sunt false:

- A. Este imposibil rezolvarea acestei probleme intr-o complexitate liniara.
- B. O varianță de rezolvare ar fi să inițiem o parcurgere spre dreapta începând din fiecare element al sirului și încercarea de a identifica urmatorul element mai mare.
- C. Cea mai bună soluție existentă implica o complexitate patratică.
- D. Nicio afirmație nu este corectă.



9. Se da urmatoarea expresie logica: $(A \text{ SI } B) \text{ SAU } (\text{NOT } B \text{ SI } C) \text{ SAU } ((\text{A} \text{ SI } \text{NOT } B) \text{ SI } C)$. Precizati pentru ce valori ale lui A, B si C, expresia are valoare de adevar TRUE

- A. A - TRUE, B - TRUE, C - FALSE
- B. A - TRUE, B - FALSE, C - FALSE
- C. A - FALSE, B - FALSE, C - FALSE
- D. A - TRUE, B - FALSE, C - TRUE

10. Precizati care dintre urmatoarele expresii logice verifică corect dacă variabila N are ultima cifra 0 sau 5:

- A. $(N \% 6 == 0 \ \&\& N \% 10 != 6) \ || \ (N \% 5 == 0 \ \&\& N \% 2 == 1)$
- B. $(N \% 10 == 0 \ || \ N \% 15 == 0)$
- C. $((N \% 10) \% 3 == 2 \ \&\& N \% 4 == 1) \ || \ (N \% 2 == 0 \ \&\& N \% 5 == 0)$
- D. $(N \% 10 == 0 \ || \ N \% 10 == 5)$



11. Domnul Gigel, a plecat în excursie împreună cu Capra, Varza și Lupul sau. Ajuțați-l să treacă raul cu cei 3 camarzi de expediție, dar se vede în față unei mari probleme: să nu treacă în același timp și lupul și capra. În plus, trebuie să nu ramane singuri pe o parte a raului, căci lupul ar mânca capra. Nici capra nu poate ramane singură cu varza, deoarece capra ar mânca varza. Domnul Gigel își pune acum problema următoare: "Oare care este numarul minim traversări de rau care trebuie facute în astă fel încât să trec cu bine toți cei 3 camarazi ai mei, fără ca vreunul să-l manance pe celalalt și la final toti cei 4 să fim pe cealaltă parte?". Il puteti ajuta?!

- A. 5 treceri
- B. 7 treceri
- C. 9 treceri
- D. Este imposibil să trec raul fără ca vreun camarad să aibă de suferit.

12-13.

A.

```
int nr_div(int n){
    int cnt = 0;
    for(int d = 1; d <= n; ++d)
        if(n % d == 0)
            cnt++;
    return cnt;
}
```

B.

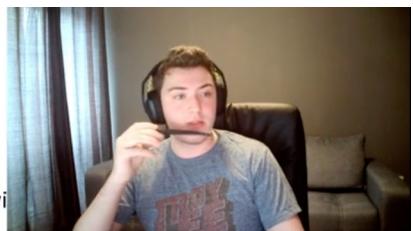
```
int nr_div(int n){
    int cnt = 0;
    for(int d = 1; d <= n; ++d)
        if(n % d == 0)
            cnt += 2;
    return cnt / 2;
}
```

C.

```
int nr_div(int n){
    int d = 2;
    int nrdiv = 1;
    while(n > 1){
        int p = 0;
        while(n % d == 0)
            p++, n /= d;
        nrdiv *= (p + 1);
        d++;
        if(d * d > n)
            d = n;
    }
    return nrdiv;
}
```

D.

```
int nr_div(int n){
    int cnt = 2;
    for(int d = 2; d * d <= n; ++d){
        if(n % d == 0)
            cnt += 2;
        if(d == n / d)
            cnt--;
    }
    return cnt;
}
```



12. Precizati care dintre cei 4 subalgoritmi calculeaza corect numarul de divizori ai unui numar pozitiv n.

- A. Algoritmii A, B si C
- B. Algoritmii A, B si D
- C. Doar algoritmii A si B
- D. Toti cei 4 algoritmi.

13. Precizati care informatii sunt adevarate referitor la complexitatile lor:

- A. Algoritmii A si B au aceiasi complexitate.
- B. Algoritmii C si D au aceiasi complexitate.
- C. Algoritm C este cel mai rapid.
- D. Toate informatiile sunt corecte.

```
bool ceFace(int n){
    int s = 1, d = n;
    while(s <= d){
        int m = (s + d) / 2;
        if(m * m * m == n)
            return 1;
        else if(m * m * m > n)
            d = m - 1;
        else s = m + 1;
    }
    return 0;
}
```

```
bool ceFace(int n){
    int i = 1;
    while(i * i * i < n)
        i++;
    return i * i * i == n;
}
```



14. Cele 2 sechete de cod:
- A. Calculeaza radical de ordin 3 din n.
 - B. Verifica daca n este cub perfect.
 - C. Utilizeaza 2 algoritmi de complexitati diferite.
 - D. Au complexitati similare.

```
int f1(int n){
    if(n == 0)
        return 0;
    int c = n % 10;
    if(c % 2 == 0)
        return f1(n / 10) * 10 + c;
    return f1(n / 10);
}
```

```
int f2(int n){
    if(n == 0)
        return 0;
    int c = n % 10;
    if(c % 2 == 0)
        return f2(n / 10) * 100 + c * 10 + c;
    return f2(n / 10) * 10 + c;
}
```



15. Se dau cele 2 functii f1 si f2. Precizati care dintre urmatoarele afirmatii sunt adevarate:

- A. f1(12345) -> 124, f2(12345) -> 1223445
- B. f1(1) -> 1, f2(1) -> 1
- C. f2(1357) -> 11335577, f1(12) -> 2
- D. Algoritmul f1 elimina toate cifrele impare, iar algoritmul f2 dubleaza toate cifrele pare.



16. Se considera o matrice de $n \times m$ elemente numere naturale. Se doreste stocarea acestor matrici ca vector. Pentru a se face acest lucru, vom folosi un sistem de asociere a valorilor din matrice, cu valorile din vectorul asociat. Ex. $A[1][1] \Rightarrow V[1]$, $A[1][m] \Rightarrow V[m]$, $A[2][2] \Rightarrow V[m + 2]$,

Dandu-se aceasta metoda de tranzitie, trebuie pentru un element din matrice sa deduceti cei 4 vecini ai sai, sus, jos, stanga, dreapta, ca si coordonate pe vectorul V. De exemplu, daca vi se da $A[i][j]$, trebuie sa deduceti pozitiile in vectorul V a elementelor $A[i-1][j]$, $A[i+1][j]$, $A[i][j-1]$, $A[i][j+1]$.

Precizati care dintre urmatoarele afirmatii sunt corecte:

- A. $A[i][j] \rightarrow SUS - V[(i-2)*m + j]$, $JOS - V[i * m + j]$, $STANGA - V[(i-1)*m + j - 1]$ – DREAPTA – $V[(i-1)*m + j]$
- B. $A[i][j] \rightarrow SUS - V[(i-1)*m + j]$, $JOS - V[(i+1) * m + j]$, $STANGA - V[i*m + j - 1]$ – DREAPTA – $V[i*m + j + 1]$
- C. $A[i][j] \rightarrow SUS - V[(i-2)*m + j]$, $JOS - V[i * m + j]$, $STANGA - V[(i-1)*m + j - 1]$ – DREAPTA – $V[(i-1)*m + j + 1]$
- D. $A[i][j] \rightarrow SUS - V[(i-1)*m + j]$, $JOS - V[(i+1) * m + j]$, $STANGA - V[i*m + j - 1]$ – DREAPTA – $V[i*m + j]$



```
int suma_speciala(int n){
    if(n == 1)
        return 1;
    return n + n * suma_speciala(n - 1);
}
```

17. Precizati ce calculeaza functia suma_speciala(n) unde n este un numar natural ≥ 1 .

- A. $n!$
- B. $n + n!$
- C. $n!/0! + n!/1! + n!/2! + \dots n!/(n-1)!$
- D. $\sum_{k=1}^n \frac{n!}{(k-1)!}$

```
void f(int n){
    for(int i = 1; i <= n; ++i){
        int k = i;
        for(int j = 1; j <= k; ++j)
            k /= 2;
    }
}
```



18. In ce clasa de complexitate se incadreaza algoritmul alaturat?

- A. $O(n * \log(n))$
- B. $O(\log(n))$
- C. $O(\log(n!))$
- D. $O(n^2)$



```
int f(int n){    o   z
    int F[11]={0,0,0,0,0,0,0,0,0,0,0};
    int a[11], nr_c = 0, m = 0;
    while(n){
        int x = n % 10;
        F[x] = 1;
        n /= 10;
    }
    for(int i = 1; i <= 9; ++i)
        if(F[i] == 0)
            a[++nr_c] = i;
    for(int i = nr_c; i >= 1; --i)
        m = m * 10 + a[i];
    return m;
}
```

19. Se da secventa de cod de mai sus. Care dintre urmatoarele afirmatii sunt corecte:
- A. Secventa alaturata utilizeaza un algoritm de sortare prin metoda numararii.
 - B. Subalgoritmul calculeaza cel mai mare numar care se poate obtine din cifrele care apar in n.
 - C. Subalgoritmul calculeaza cel mai mare numar care se poate obtine din cifrele distincte care nu apar in n.
 - D. Subalgoritmul calculeaza cel mai mare numar care se poate obtine utilizand toate cifrele distincte nenule care nu apar in n.



```
void f(int n){
    int p = 1;
    while(n > 0){
        if(n % 2 == 1)
            cout << p << " ";
        p = p * 2;
        n /= 2;
    }
}
```

20. Ce face secenta de cod alaturata?

- A. Il scrie pe n in baza 2 si afiseaza pozitiile bitilor egali cu 1.
- B. Determina termenii sumei puterilor lui 2 care au ca rezultat pe n.
- C. Il imparte pe n la 2 pana cand nu il mai poate imparti si la fiecare pas, daca n este impar, afiseaza 2 la puterea pasului -1.
- D. Se afiseaza scrierea lui n in baza 2.

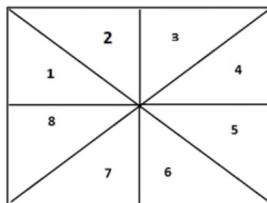


21. Fie o matrice pătratică cu zonele **1,2,3,4,5,6,7,8** ca în figura următoare.

elementele matricii aflate pe cele două diagonale și pe cele 2 axe de simetrie.

Atenție: dacă **n** este par cele două axe de simetrie nu conțin nici un element impar atunci

axe de simetrie conțin elemente ale matricii. Care afirmații sunt adevărate?



- A. Elementele din zona 1 sunt în număr de $(n^2 - 4*n)$ DIV 8, pentru $n > 1$;
- B. Elementele din zona 2 sunt în număr de 3 elemente, pentru $n = 7$;
- C. Elementele din zona 3 sunt în număr de $(n^2 - 2*n*(1+(n \text{ MOD } 2)) + 2*(n \text{ MOD } 2))$ DIV 8, $n > 1$;
- D. Elementele din zona 4 sunt în număr de $(n^2 - 2*n*(1+(n \text{ MOD } 2)) + 3*(n \text{ MOD } 2))$ DIV 8, $n > 1$.

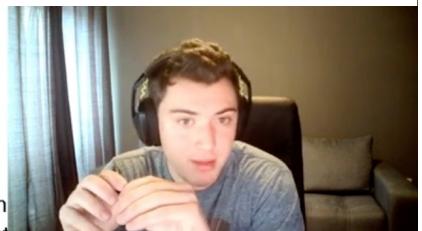


22. Pentru afisarea unei clepsidre de dimensiune n , se utilizeaza caractere * si # urmatorul (pentru $n = 5$ se afiseaza):

```
*****
#*****
###**#
###**#
###**#
###**#
#*****
*****
```

Utilizand acest mod de generare, precizati care dintre urmatoarele afirmații sunt adevărate:

- A. Pentru $n = 7$, “*” – 98 , “#” – 72
- B. Pentru $n = 10$, “*” – 199, “#” – 162
- C. Pentru $n = 1$, “*” – 1, “#” – 1
- D. Pentru $n = 7$, “*” – 99, “#” – 72



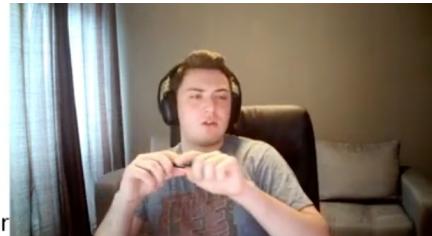
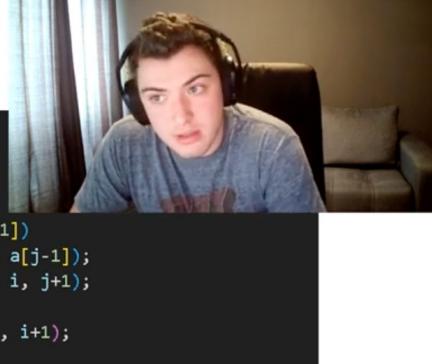
23. Cei n participanți la o competiție sunt organizați în m echipe, fiecare participant fiind singură echipă. În timpul competiției fiecare participant dintr-o echipă devine prieten cu toate ceilalăii participanți din aceeași echipă. Dându-se m și n aflați numărul maxim și numărul minim de prietenii potențiale. Exemplu $n = 6$ participanți și $m = 3$ echipe \Rightarrow Max=6 pentru distribuția 1,1,4 iar Min=3 pentru distribuția 2,2,2.

- A. $n = 11, m = 3 \rightarrow \text{MAX} = 36, \text{MIN} = 15$.
- B. $n = 12, m = 4 \rightarrow \text{MAX} = 36, \text{MIN} = 10$.
- C. $n = 50, m = 20 \rightarrow \text{MAX} = 190, \text{MIN} = 30$.
- D. $n = 15, m = 6 \rightarrow \text{MAX} = 45, \text{MIN} = 12$

```
void ce_face1(int a[], int n, int i = 1, int j = 1){  
    if(i == n)  
        return ;  
    if(j <= n){  
        if(a[i] > a[j])  
            swap(a[i], a[j]);  
        ce_face1(a, n, i, j+1);  
    }  
    else ce_face1(a, n, i+1, i+1);  
}
```

```
void ce_face2(int a[],  
             int n)  
{  
    if(i == n)  
        return ;  
    if(j <= n){  
        if(a[j] < a[j-1])  
            swap(a[j], a[j-1]);  
        ce_face2(a, n, i, j+1);  
    }  
    else ce_face2(a, n, i+1);  
}
```

```
void ce_face3(int a[], int n, int i = 1, int j = 1){  
    if(i == n)  
        return ;  
    if(j >= 2){  
        if(a[j] < a[j-1])  
            swap(a[j], a[j-1]);  
        ce_face3(a, n, i, j-1);  
    }  
    else ce_face3(a, n, i+1, i+1);  
}
```



24. Considerand cele 3 functii din imaginea de mai sus, trebuie sa selectati care sunt corecte:

- A. Subalgoritmul ce_face1 sorteaza utilizand metoda de sortare SelectionSort, descrescator
- B. Subalgoritmul ce_face1 si ce_face2 utilizeaza aceiasi metoda de sortare si sorteaza crescator.
- C. Subalgoritmul ce_face1 sorteaza prin SelectionSort, iar ce_face3 sorteaza prin InsertionSort.
- D. Toti algoritmii sorteaza crescator sirurile de numere, utilizand algoritmi de sortare diferiti.

Barem Januarije 2023

1.	BC
2.	A
3.	BC
4.	BC
5.	BD
6.	AB
7.	A
8.	ACD
9.	ABD
10.	D
11.	B
12.	A
13.	AB
14.	BC
15.	D
16.	C
17.	CD
18.	AC
19.	AD
20	BC
21.	BD
22.	B
23.	AD
24.	CD