

Răspunsurile corecte cu explicații

1. a

Pentru calculul mediei aritmetice, este necesară calcularea sumei elementelor vectorului după care se împarte rezultatul la numărul elementelor. Dintre variantele de răspuns, numai una (varianta a) face acest lucru, celelalte variante fie împart sumele parțiale la numărul elementelor, fie adună indexul elementelor la suma și nu valoarea elementelor stocate în vector.

2. d

Întrucât instrucțiunea `while` nu testează dacă `x` este egal cu `y` ci îi atribuie lui `x` valoarea `y`, care este un număr diferit de 0, va fi asimilat ca un test adevărat, dar `x` va avea valoarea 15. Cum testul `15 > 15` impune intrarea pe ramura `else`, `y` va deveni 0. La următorul test în `while` `x=y=0`, 0 este asimilat ca false, deci se părăsește bucla. Se afișează `x` care are valoarea 0.

3. d

Pentru secvența de cod dată și pentru datele de intrare specificate, se obține același rezultat dacă se folosește instrucțiunea `while` sau `do ... while`.

4. c

Instrucțiunea `for` inițializează variabilele `i` și `s` cu valoarea 0. Instrucțiunea `while` executată pentru `i = 0` se va executa de 5 ori, pentru valorile 1, 2, 3, 4, 5. Se intră în `while` și pentru `i = 5` pentru că, la testul `i++ < 5`, se compară cu 5 valoarea inițială a lui `i` (în acest caz 4) și abia după aceea se incrementează. După executarea instrucțiunii `while` `i` va avea valoarea 5, deci nu se va mai executa un nou pas al instrucțiunii `for`.

5. b

Prima instrucțiune `for` se execută de 3 ori, pentru `i = 3, 2, 1`. Pentru `i = 3`, a doua instrucțiune `for` se va executa de 4 ori, pentru `j = 3, 2, 1, 0`. Pentru fiecare dintre aceste valori, se va afișa caracterul corespunzător codului ASCII rezultat din adunarea codului ASCII a caracterului 'A' cu valoarea variabilei `j`. Deci, pentru `i = 3`, se vor afișa caracterele DCBA. Se repetă raționamentul pentru celelalte valori ale lui `i`, pentru care se vor afișa CBA și BA.

6. a

Se folosește definiția operațiilor de incrementare/decrementare pre și post fixate. Pentru incrementarea prefixată (`++i`) se incrementează variabila și apoi se folosește, incrementarea postfixată (`i++`) folosește variabila și apoi o incrementează.

7. c

8. a

Codul furnizat pune 0 în locul elementelor aflate pe pozițiile care împărțite la 3 dau restul 0. Așadar, răspunsul corect este 0 4 5 0 2 4 0 7.

9. b

Funcția calculează suma elementelor de pe linia și de pe coloana dată de parametrul `val`. Deci, se aduna elementele de pe linia 2 (suma lor este 15) și de pe coloana 2 (suma este tot 15), deci răspunsul este 30.

10. a

Funcția `f` calculează suma elementelor de pe coloana indicată ca al treilea parametru al funcției, iar procedura *Algorithm* apelează funcția și determină valoarea maximă dintre valorile returnate de funcție.

11. c

```
// Example program
#include <iostream>
#include <string>
using namespace std;

int main()
{
    int i, j, a[10][10], n=4;
    for (i=1; i<=n; i++)
        for (j=1; j<=n; j++)
        {
            a[i][j] = (i+j-1)%n;
            a[j][i] = a[i][j];
        }
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=n; j++)
            cout<<a[i][j]<<" ";
        cout<<endl;
    }
    return 0;
}
```

12. b

Interschimbare 4 cu 1, interschimbare 7 cu 3, interschimbare 7 cu 5

13. b

14. c

Din lista de şiruri de caractere se scot două cu lungimile cele mai scurte, se concatenează şi rezultatul concatenării este pus înapoi în listă. Procesul se repetă până în listă este un sigur cuvânt. Concatenările care se efectuează sunt: *mere+prune*, *ananas+smochine*, *mereprune+portocale*, *clementine+ananas+smochine*, *merepruneportocaleclementineananas+smochine*.

15. b

Funcţia f utilizează algoritmul backtracking pentru a genera toate submulţimile mulţimii $\{1, 2, 3, \dots, n\}$. Se vor genera C_n^k submulţimi de k elemente, astfel, numărul total de submulţimi generate este $\sum_{k=1}^n C_n^k = 2^n - 1$. Pentru $n = 6$ se vor genera 63 de submulţimi.

16. c

Pentru decodificarea mesajului este necesar să urmărim paşii algoritmului de codificare în ordine inversă:

- se împarte mesajul în fragmente;
- se construieşte tabelul de codificare;
- se determină ordinea de parcurgere a coloanelor;

S	U	B	I	E	C	T
A	D	M	I	T	E	R
E		L	A		I	N
F	O	R	M	A	T	I
C	A					

17. c

c) Si P şi Q Q este adev: Graful este neorientat, deci orice muchie creşte suma gradelor cu 2. P este adev: dacă se consideră suma gradelor şi se scad toate gradele pare rezultă un număr par (q este adev). Prin urmare, numărul de grade impare este par.

18. c

Fiecare nod din primul set este conectat cu fiecare nod din al doilea set. Prin urmare, numărul total de muchii este $m * n$.

19. a

Primul număr va fi rădăcina, iar următoarele $N/4 - 1$ numere vor fi în subarborele drept. Celelalte $3 * N/4$ numere vor ajunge în subarborele stâng și, deoarece sunt introduse în ordine descrescătoare, fiecare număr va fi adăugat ca fiu stâng al ultimului nod din stânga. Astfel, calea cea mai lungă pornește de la rădăcină și traversează cele $3 * N/4$ noduri care au fost introduse în ordine descrescătoare. Deci, numărul de legături între cele $3 * N/4 + 1$ noduri este $3 * N/4$.

20. b

Numărul de muchii dintr-un graf turneu cu n noduri este același cu numărul de muchii dintr-un graf complet cu n noduri. În plus, la un graf turneu, fiecare muchie are o orientare. Pentru un graf complet cu n noduri, numărul de muchii este $n * (n - 1)/2$. Dacă fiecare muchie are o orientare atașată (există două orientări posibile pentru fiecare muchie), se poate calcula numărul de posibilități în care aceste muchii pot fi aranjate. Cu alte cuvinte, avem $n * (n - 1)/2$ sloturi (reprezentate de muchii), fiecare slot putând lua două valori posibile (reprezentate de cele două orientări posibile). Astfel, numărul total de posibilități este $2^{n * (n - 1)/2}$.

21. b

Graful respectiv este defapt un arbore binar (în care fiecare nod are maxim doi fii) cu rădăcina 1 și toate nivelurile complete mai puțin ultimul nivel care este umplut peste jumătate. Înălțimea arborelui este $\log_2 100 = 6$, deci, distanța de la rădăcina 1 la frunza 64 este 6. Pe ultimul nivel al arborelui sunt 37, deci nodul 100 este în subarborele drept (raportat la rădăcină), iar distanța de la nodul 1 la 100 este tot 6. Astfel, drumul cel mai lung este între nodurile 64 și 100 pentru că 64 și 100 sunt pe ultimul nivel, iar drumul trece prin rădăcină.

22. d

Funcția calculează numărul de apeluri recursive până când cele două argumente au aceeași valoare la care se adună de două ori valoarea la care s-a ajuns. Valoarea lui x crește cu 1 la fiecare apel, iar valoarea lui y scade cu 1 la fiecare apel, întâlnindu-se la valoarea $(x + y)/2 = 1250$. Deci, rezultatul este egal cu $1250 + 1250 + 250 = 2750$ pentru că au fost 250 de apeluri recursive (la fiecare apel s-a adunat 1 la rezultat).

23. a

Funcția recursivă calculează numărul de cifre dintr-un număr întreg.

24. a

a) Plină: $(\text{ULTIM}+1) \bmod n = \text{PRIM}$, Goală: $\text{ULTIM} = \text{PRIM}$

```
typedef int atom;
#define dimmax 8
struct queue

    int head, tail;
    atom vect[dimmax];

t nextpos(int index);
t isempty(queue Q);
id initQ(queue &Q);
id put(queue &Q, atom a);

t nextpos(int index)

    if (index < dimmax - 1)
        return index + 1;
    else return 0;
```

```

t isempty(queue Q)

    return(Q.head == Q.tail);

id initQ(queue &Q)

    Q.head = Q.tail = 0;

id put(queue &Q, atom a)

    //if (Q.head == nextpos(Q.tail))
    if ((Q.tail + 1)% dimmax == Q.head) //asta se testeaza
        cout << "\n buffer plin!";
    else
    {
        Q.vect[Q.tail] = a;
        Q.tail = nextpos(Q.tail);
    }

in main
eue Q;
    initQ(Q);
    for (int i = 1; i <= 4; i++)
    {
        put(Q, i);
        cout << Q.tail << " " << Q.head << endl;
    }
    for (int i = 1; i <= 6; i++)
    {
        cout << "extrag " << get(Q) << endl;
    }
    cout << Q.tail << " " << Q.head << endl;
    for (int i = 1; i <= 9; i++)
    {
        put(Q, i);
        cout << "pun " << i << " ";
        cout << Q.tail << " " << Q.head << endl;
    }

```

25. a

Varianta I reprezintă rezolvarea recursivă, iar varianta IV este rezolvarea nerecursivă a problemei.

Dobânda este de 0,5% pe an, adică este $0.5/100$ pe an, iar pentru o lună dobânda este $0.5/100/12$. După prima lună Alecu va avea în cont s (suma inițială) la care se adaugă dobânda pe prima lună, adică $s * 0.5/100/12$. Suma totală la sfârșitul primei luni va fi deci $s * (1 + 0.5/100/12)$. După 2 luni suma din cont va fi $s * (1 + 0.5/100/12) * (1 + 0.5/100/12)$ s.a.m.d.