

Răspunsurile corecte cu explicații

1. d

Subexpresiile care reprezintă termenii sumei extrag câte o cifră zecimală din valoarea variabilei *i*. Prima subexpresie are valoarea 5, iar cea de a doua are valoarea 3. Suma valorilor este 8.

```
void zoom(int &m, int img[][DIMMAX])

int imNoua[DIMMAX][DIMMAX];
int iNou = 0;
int jNou = 0;
for (int i = 0; i < m; i++)
{
    jNou = 0;
    for (int j = 0; j < m; j++)
    {
        imNoua[iNou][jNou] = img[i][j];
        imNoua[iNou][jNou + 1] = img[i][j];
        imNoua[iNou + 1][jNou++] = img[i][j];
        imNoua[iNou + 1][jNou++] = img[i][j];
    }
    iNou += 2;
}
m *= 2;
for (int i = 0; i < m; i++)
{
    for (int j = 0; j < m; j++)
    {
        img[i][j] = imNoua[i][j];
    }
}
```

3. c

Funcția `strcpy` copie în șirul *d* porțiunea "ere2022" din șirul *s*, indicată prin expresia *s*+5. Apelul funcției `strcpy` returnează adresa șirului *d*. Această adresă este utilizată la apelul funcției `strcat` care concatenează la finalul caracterelor șirului *d* porțiunea "2022" a șirului *s*, indicată prin expresia *s*+8.

4. d

Valoarea lui *a* se modifică în funcție, ultima valoare fiind primită în return.

5. b

Codul este

```
int n = 1234;
int p = 1;
int m = n % 10;
n = n / 10;
int q = 1;
while (n != 0) {
    q = q + 1;
    for (int i = 1; i <= q; i++)
    {
        p = p * 10;
    }
}
```

```

    m = (n % 10) * p + m;
    n = n / 10;
}
cout << "m=" << m << endl;

```

6. d

Se verifica usor

7. b

①	1	①	1	0
1	1	1	0	1
①	0	1	①	1
1	1	1	1	①
1	0	①	0	1

8. b

Funcția f parcurge elementele matricii TA până la linia N-1 și coloana N-1. Pentru fiecare element se verifică valorile vecinilor ortogonal adiacenți (și valizi). Dacă cel puțin un vecin are aceeași valoare cu elementul curent, atunci se incrementează valoarea ce va fi returnată de către funcție. Pentru valorile parametrilor din enunț, funcția va returna numărul de elemente din întreaga matrice cu valoarea 'C' care au măcar un vecin ortogonal cu aceeași valoare.

9. c

Dacă se scrie polinomul sub forma: $P(x) = a_0 + x(a_1 + x(a_2 + \dots x(a_{n-1} + a_n x) \dots))$ se vor efectua doar n operații de înmulțire.

10. b

n trebuie să fie divizibil cu 6, astfel tripletul este de forma $n = n/6 + n/3 + n/2$

$$6=1+2+3$$

$$12=2+4+6$$

$$18=3+6+9$$

$$24=4+8+12$$

$$30=5+10+15$$

$$36=6+12+18$$

$$42=7+14+21$$

$$48=8+16+24$$

$$54=9+18+27$$

$$60=10+20+30$$

$$66=11+22+33$$

$$72=12+24+36$$

$$78=13+26+39$$

$$84=14+28+42$$

$$90=15+30+45$$

$$96=16+32+48$$

11. d

Sortare crescătoare cu QuickSort, cazul cel mai defavorabil. Apel de QuickSort pe 5 elemente, generează 4 apeluri de QS, respectiv 1 apel de P. Apel de QuickSort pe 10 elemente, generează 9 apeluri de QS, respectiv 4 apeluri de P. Pe n elemente, n-1 apeluri de QS și $n/2 - 1$ apeluri de P, $n = 1000 \rightarrow$ P:499 apeluri, QS: 999 apeluri

12. a

13. a

Programul generează siruri de biți de lungime n . $n=2 \cdot 2^2 = 4$ *siruri* $= 3 \cdot 2^3 = 8$ *siruri* $= 4 \cdot 2^4 = 16$ *siruri* *generalizăm* $= 1024$ *0* $= 1024$ *siruri*

14. b

Se consideră număratoarea în cerc ca o număratoare liniară în mai multe siruri mici, fiecare cu n cicliști, obținând un sir mare cu p cicliști (p fiind multiplu de n). Număratoarea se termină atunci când al n -lea ciclist (dintr-un sir mic) este selectat, și astfel, următorul ciclist care ar trebui să fie selectat va fi un al k -lea din următorul sir mic. Mai exact, p trebuie să fie și multiplu de k , adică $p = \text{cmmmc}(n, k)$. Dintre cei p cicliști, au fost selectați exact p / k cicliști, deci sportivii neselectați vor fi în număr de $nr = n - p/k = n - \text{cmmmc}(n, k)/k = n - (n \cdot k / \text{cmmdc}(n, k))/k = n - n/\text{cmmdc}(n, k)$. Vor fi selectați toți dacă $\text{cmmdc}(n, k) = 1$.

15. c

a) Are doar o soluție: Apa Aer Foc Vant Pamant b) A treia soluție este Apa Aer Foc Vant Lemn c) A treia soluție este Aer Foc Pamant Metal Lemn d) A treia soluție este Apa Foc Vant Pamant În Validare se observă că se impune o ordine între soluții!

```
#include <iostream>
using namespace std;

int x[10], n=7, p=5;
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; // sau 0
}

void Afisare(int k)
{
    for(int i=0; i<p; i++)
        cout<<magic[x[i]]<<" ";
    cout<<endl;
}

void Back(int k){
    for(int i = 0 ; i < n ; ++i) {
        x[k]=n-1-i;
        if( Validare(k) )
            if(k>=p-1) Afisare(k);
        else
            Back(k+1);
    }
}

int main(){
    Back(0);
    return 0;
}
```

16. d

Frunzele sunt reprezentate de nodurile care nu apar în vectorul de "tați". Așadar, nodurile 1, 4, 5, 6, 10, 12 reprezintă frunzele arborelui dat.

17. a

Pentru a acoperi cele 34 de muchii sunt necesare 9 noduri, aşadar restul de 51 de noduri au gradul 0.

18. c

Graful iniţial are $5 \cdot 4 / 2 = 10$ muchii. Dacă eliminăm: - un singur nod vom obţine 4 noduri şi 6 muchii - două noduri, vom obţine 3 noduri şi 3 muchii - 3 noduri, vom obţine 2 nod şi 0 muchie - 4 noduri, va ramane 1 nod şi 0 muchii

19. b

100 de valori nenule în matricea de adiacenţă înseamnă 50 muchii. $n(n-1)/2 = 50 \Rightarrow n = 11$ noduri pt o componentă cât mai completă, deci 11 noduri pot face parte dintr-o componentă conexă şi $2022 - 11 = 2011$ noduri izolate, adică 2012 componente conexe.

20. b

daca notam cu nr nr de noduri a unei componente conexe, înseamnă că numărul de muchii al acesteia este nr-1 (altfel nu ar fi arbore). Cum sunt 10 comonente conexe, numarul total de muchii va fi $40 - 10 = 30$.

21. b

Se poate construi arborele de parcurgere a spatiului solutiilor pornind de la nodul cu suma totala si ramificand prima data dupa mestecan (pot fi 2, 1 sau 0), dupa aceea pentru arbusti iar de restul banilor se cumpara trandafiri ca sa nu ramana rest mai mare ca 9:

2 mesteceni $\rightarrow$ 1 sau 0 arbusti (2 variante)

1 mestecan $\rightarrow$ 3, 2, 1, 0 arbusti (4 variante)

0 mesteceni $\rightarrow$ 5, 4, 3, 2, 1, 0 arbusti (6 variante)

In total 12 variante.

22. c

Pentru a acoperi cele 34 de muchii sunt necesare 9 noduri, aşadar restul de 51 de noduri au gradul 0.

23. b

Funcţia parcurge şirul de numere întregi în sens descrescător începând cu indexul 4. Valoarea returnată după fiecare apel recursiv este dată de suma elementului curent (indicat prin valoarea indexului i) şi valoarea returnată de următorul apel multiplicată cu 10. Se obţine astfel un număr alcătuit din cifrele/valorile tabloului a.

24. c

Funcţia realizează parcurgerea şirului de numere întregi în sens descrescător începând cu indexul 4. Înainte de fiecare apel recursiv al funcţiei, valoarea elementului curent (indicat prin valoarea indexului i) este actualizată prin sumarea la valoarea sa a valorii elementului anterior. Elementele ale căror valori sunt modificate sunt cele de pe poziţiile (indecşii) 1, 2 şi 4. Valorile lor devin 96, 16 şi respectiv 57.

25. b

Se returneaza indexul, nu valoarea din vector, pentru poziție de la 1 la 4.