Răspunsurile corecte cu explicații

1. b

Valoarea de adevăr a condiției instrucțiunii while este 0 (zero) motiv pentru care cuvântul Queen va fi afișat o singura dată.

2. c

3. c

f1 și f2

4. a

Deși vectorul are 10 elemente, la apel n va avea valoarea 6. Cum bucla pentru merge doar până la 3, prima interschimbare va fi între valoarea de pe poziția 1 și 5 (6-1), apoi 2 și 4, respectiv 3 cu 3. Deci la a doua buclă pentru vor fi afișate valorile 5 4 3 2 1 6 (ultima valoare nu se interschimbă)

5. a

Trebuie căutate în matrice valorile care au deasupra și dedesubt valori mai mici cu 1 (modulo 3 - număr pozitiv) și la stânga și dreapta valori mai mari cu 1 (modulo 3).

2	0	0	0	0	2	0
2	2	(1)	2	(1)	2	2
0	1	0	0	0	2	2
2	2	0	1	1	1	2
1	(0)	1	0	0	(2)	0
2	2	2	(1)	2	1	2
1	0	1	0	1	2	1

6. d

Comanda return forțează ieșirea din funcție înainte de a fi executată și instrucțiunea de după apel. Mai mult, este necesar a se observa ca adunarea din return nu are efect asupra valorilor de afișat. Astfel, prima valoare afișată este valoarea lui n (adică 10), apoi 9 8 7 6 5 4 3 2 1. Pentru n=0, se intră pe ramura adevărat a funcției și se afișează valoarea 0. Toate afișările sunt effectuate doar de instrucțiunea cout dinainte de return n+funcție(n-1).

7. b

În situația în care inițializarea elementelor unui tablou nu este completa (nu sunt precizate valorile tuturor membrilor tabloului) acestea din urmă vor fi inițializate cu valoarea 0.

8. d

Cele trei puncte sunt situate în coordonatele (3,3), (1,4), (3,1) - notate cu **x** în matricea de mai jos.

0	0	0	x
1	0	1	1
x	0	x	1
1	0	0	0

Drumul de lungime minimă este:

- de la (1,1) la (1,4) (lungime 3)
- de la (1,4) la (3,1) (lungime 5)
- de la (3,1) la (3,3) (lungime 2)
- de la (3,3) la (4,4) (lungime 2)

Lungime totală: 3 + 5 + 2 + 2 = 12.

9. a

După primele două operații de ADD matricea rezultată este:

8	0	0	0	0
0	0	**10**	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

SUM(1,3) => suma elementelor încercuite: **8**

După următoarele două operații de ADD matricea rezultată este:

8	0	0	0	0
0	**4**	**10**	0	0
0	**6**	0	0	0
0	0	0	0	0
0	0	0	0	0

SUM(3,2) => suma elementelor încercuite: **18**

UNDO(2) => anulează ultimele 2 operații de ADD După următoarea operație de ADD matricea rezultată este:

15	0	0	0	0
0	0	**10**	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

SUM(3,2) => suma tuturor elementelor încercuite: **25** UNDO(3) => anulează mai întâi ultima operație de ADD, apoi efectul operației UNDO(2) și încă o operație de ADD

8	0	0	0	0
0	0	**10**	0	0
0	**6**	0	0	0
0	0	0	0	0
0	0	0	0	0

SUM(4,4) => suma tuturor elementelor încercuite: **24**

10. c

```
int a[10], i, smax, sc, p, pc, u;
 for (i = 0; i < 10; i++)
     cin >> a[i];
 smax = a[0]; sc = a[0]; p = 0; u = 0; pc = 0;
 for (i = 1; i <= 9; i++)
 {
     if (sc > 0) sc += a[i];
     else
     {
         sc = a[i]; pc = i;
     }
     if (sc > smax)
     {
```

```
            p = pc; u = i; smax = sc;
        }
    }
    for (i = p; i <= u; i++)
        cout << a[i] << " ";
    cout << endl;
    cout << "max= " << smax << endl;
```

11. d

(int)(sqrt(n)).

12. c

Funcția calculează h*suma a n elemente, unde $h = log(n)$.

13. b

Comparațiile se realizează cu elementele 23, 66 și 70.

14. b

Regina 3 poate fi plasată doar după ce se face pasul înapoi şi se replasează regina 2 (Figura (b)). Celelalte figuri reprezintă mai multe operaţii.

15. b

Se calculează suma tuturor elementelor S. => O(n) Se consideră două sume S1=0 suma elementelor dinainte pe poziția 1 și S2= S-a[1]. Într-o singură trecere, se compara S1 cu a[i] și cu S2, se decide dacă a[i] este corespunzător sau nu, apoi, înainte de a se trece la pasul i+1, se recalculează S1=s1+a[i], S2=S2-a[i]. Deci se poate crea un algoritm de complexitate O(n).

16. a

Se caută în matricea M liniile care au doar valori de 0. Toate aceste linii corespund sarcinilor care nu au dependențe și pot fi duse la îndeplinire în acest punct. Apoi, se scot din matricea M liniile respective și coloanele corespunzătoare sarcinilor respective. Procesul se repetă până nu mai sunt sarcini de îndeplinit, sau există o dependență circulară între sarcini (caz în care nu pot fi îndeplinite toate sarcinile).

17. a

Un graf conex n vârfuri este arbore dacă și numai dacă este conex și are $n-1$ muchii. Notăm cu cu n_i numărul vârfurlor și cu m_i numărul muchiilor componentei conexe $C_i, i = 1, \ldots, k$. Avem $n = \sum_{i=1}^{i=k} n_i$ și $m = \sum_{i=1}^{i=k} m_i$. Din componenta C_i trebuie eliminate $m_i - n_i + 1$. Rezultă că numărul muchiilor eliminate va fi $\sum_{i=1}^{i=k} (m_i - n_i + 1) = m - n + k$.

18. b

Evident, o *groapă* i poate exista dacă $A = \{(j, i)/j \neq i\}$. Presupunem că ar exista două *gropi* i_1 și i_2. i_1 fiind *groapă*, arcul $(i_2, i_1) \in A$ și arcul $(i_1, i_2) \notin A$. Similar, i_2 fiind *groapă*, arcul $(i_1, i_2) \in A$ și arcul $(i_2, i_1) \notin A$. Imposibil.

19. a

Se demonstreaza prin inducție după n că numărul nodurilor complete este egal cu numărul frunzelor minus 1

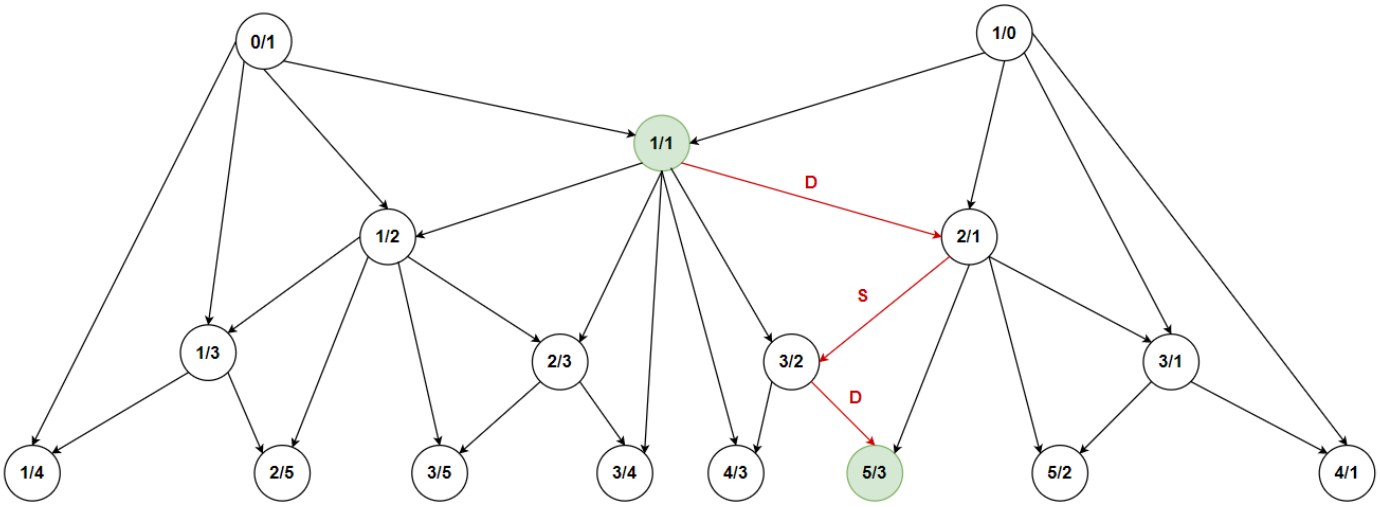
20. b

Pentru m = 3 și n = 5 se obține o rețea care are:

- pe nivelul 0 nodurile: 0/1 și 1/0
- pe nivelul 1 nodul: 1/1
- pe nivelul 2 nodurile: 1/2 și 2/1
- pe nivelul 3 nodurile: 1/3, 2/3, 3/2 și 3/1
- pe nivelul 4 nodurile: 1/4 , 2/5, 3/5 , 3/4, 4/3 , 5/3, 5/2 și 4/1

Fracția căutată se află pe nivelul 4, drumul parcurs din nodul 1/1 până la aceasta se codifică astfel: **DSD**.

Rețeaua asociata este cea din figura de mai jos:



21. d

Se desenează graful și, vizual, se determină drumul minim de la toate nodurile i la toate nodurile j, unde i < j. Rezultatul este maximul valorilor rezultate.

O altă variantă de resolvare este construirea matricei de costuri și aplicarea algoritmului Floyd–Warshall de determinare a drumului minim între oricare două vârfuri din graf. Rezultatul este maximul din matricea de costuri rezultată aplicării algoritmului.

Maximul drumurilor minime este 8, prin vârfurile 3-5-4.

22. b

Funcția întoarce ((1 + 2 . . . + x-1 + x) +y), adică x(x+1)/2 + y.

23. b

Deoarece r=10, funcția întoarce suma cifrelor: 3 + 4 + 5 = 12

24. c

Când se apelează funcție(4), chiar la intrearea in funcție se afișează primul semn #. Pentru că inițial n este diferit de 0, se va afișa valoarea lui n, 4 apoi numerele de la 0 la 3 din bucla for, Se apeleaza funcție(3) urmat de afișarea unui nou # (de la intrarea in funcție) , numărul 3 și apoi valorile de la 0 la 2, Se apeleaza funcție(2) se afisează #, 2 și valorile de la 0 la 1 Se apeleaza funcție(1) se afisează #, 1 și valoarea 0 Se apeleaza funcție(0) se afisează #, și pentru că neste 0, se afișează încă un # de pe ramura else. Dupa aceasta se vor afisa încă 4 semne # pentru instrcțiunea de afișare de după apelul recursiv. În total 10 semne #

25. b

Dimensiunea oricarei stive poate fi mai mare decat MAXSIZE/2. Ambele stive isi maresc dimensiunea pornind din ambele capate și dacă vârful uneia ajunge lângă vârful celeilalte, atunci stivele sunt pline. Astfel, condiția va fi top1=top2-1