Răspunsurile corecte cu explicații

1. a

Întrebarea testează cunoștințele legate de domeniul de vizibilitate al variabilelor. Un eventual apel al funcției va „ascunde" variabila globală redefinită local.

2. d

n/1000 *100=9800
n% 100 =65

3. b

4. a

5. c

6. c

```
int a, b, c, d,n, m;
cin >>a; cin>>b;
m = 0;
for (n = a;n <= b;n++)
{
    c = 0;
    for (d = 1; d <= n; d++)
    {
        if ((n % d) == 0) c++;
    }

    if (c > m)
        m = c;
}
for (n = a;n <= b;n++)
{
    c = 0;
    for (d = 1; d <= n; d++)
    {
        if ((n % d) == 0)
            c++;
    }

    if (c == m)
        cout<<"n="<<n<<endl;
}
```

7. b

Elementele din vector a21 a31 a32 a41 a42 a43 a51 ....

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 nici un vecin nu are aceeași valoare cu elementul curent, atunci se incrementează valoarea ce va fi returnată de către funcție. Pentru valorile paramentrilor din enunț, funcția va returna numărul de elemente din întreaga matrice cu valoarea 'C' care nu un vecin ortogonal cu aceeași valoare.

9. a

Interclasarea $v_1 \oplus v_2$ are complexitatea $\mathcal{O}(n+m)$ și rezultă un vector de dimensiune $n+m$. Acesta este interclasat cu vectorul v_3, iar complexitatea este $\mathcal{O}(n+m+m) = \mathcal{O}(n+m)$ și vectorul rezultat este de dimensiune $n+2\cdot m$. Acest ultim vector este interclasat cu v_4, iar complexitatea este $\mathcal{O}(n+2\cdot m+m) = \mathcal{O}(n+m)$.

10. a

1. Află min din lista de elemente 2. Swap cu valoarea de pe prima poziție 3. Repetă pasii de mai sus pentru restul listei (pornind de la pozitia următoare și avansând de fiecare data)

11. a

Numărul submulțimilor formate din k elemente este C_n^k. Rezultă că numărul submulțimilor lui M este $\sum_{k=0}^{n} C_n^k = 2^n$ (Binomul lui Newton).

12. b

Se aplică backtracking

13. b

14. b

Cele n iterații ale primei bucle sunt de complexitate: $m-1, m-2, \ldots, m-n$. Rezultă o complexitate: $n*m-(1+2+\cdots+n) = n*m-n(n+1)/n => \mathcal{O}(n\cdot(m-n))$. Al doilea for cu indicele i are complexitatea $m-n$ care mai mică decât complexitatea primului for.

15. d

Toţi algoritmul enumeraţi au complexitatea timp $O(n^2)$ în cazul cel mai nefavorabil.

16. b

Cifrele 2, 4 si 1 apar de 2 ori, deci 6 cifre cu suma 14. Celelalte cifre pentru cel mai scurt cod sunt 0, 3, 5, 6, 7 si 9. Suma va fi 30 Daca nu nu ne referim la cel mai scurt cod (cazul de la A), 8 poate fi in cod. Suma cifrelor în cel mai scurt cod posibil poate fi 38. //pt b, este 44

17. b

Se consideră numărătoarea în cerc ca o numărătoare liniară în mai multe șiruri mici, fiecare cu n spectatori, obținând un șir cu p spectatori (p fiind multiplu de n). Numărătoarea se termină atunci când al n-lea spectator (dintr-un șir mic) este selectat, și astfel, următorul spectator care ar trebui să fie selectat va fi un al k-lea din următorul șir mic. Mai exact, p trebuie să fie și multiplu de k, adică p = cmmmc(n, k). Dintre cei p spectatori, au fost selectati exact p / k, deci spectatori neselectați vor fi în număr de nr = n - p/k = n - cmmmc(n,k)/k = n - (n*k/cmmdc(n,k))/k = n - n/cmmdc(n,k). Vor fi selectați toți dacă cmmdc(n,k)=1.

18. d

s-a impus o ordine a afișării. soluțiile sunt: 1. Lopata Sapa Topor Cazma 2. Lopata Sapa Topor Grebla 3. Lopata Sapa Cazma Grebla 4. Lopata Topor Cazma Grebla 5. Sapa Topor Cazma Grebla

19. d

In orice arbore binar nr frunzelor (grad 0) este cu unu mai mare decat nr de noduri de grad 2. Prin urmare grad0=grad2+1=13. Nr de noduri de grad 1 este 31-(12+13)=31-25=6

20. a

Rădăcina este 0. Fii rădăcinei sunt valorile de la 1 la 9. Următorul nivel va conține numerele de la 10 la 99, următorul de la 100 la 999, apoi de 1000 la 9999, 10000-99999, 100000-999999. Indiferent ce valoare mai mică de un milion se caută for fi nevoie de 7 accesări de elemente pentru că atâtea sunt necesare a ajunge de la rădăcină la frunze.

21. a

Se formează două subgrafuri complete, cu noduri cu identificatori pari, respectiv impari. Fiecare subgraf are $\frac{n-1}{2}$, respectiv $\frac{n+1}{2}$ noduri. Ambele subgrafuri au în total $\frac{n^2-2n+1}{4}$ muchii. Există și câte o muchie între orice noduri cu identificatori de forma $< 2k, 2k+1 >$ (din subgraful par, respectiv impar), cu mențiunea că nodul 1 (din subgraful impar) nu formează pereche cu nici un nod din celălalt subgraf: total $\frac{n-1}{2}$ astfel de muchii. Prin urmare, numărul total de muchii este: $\frac{n^2-2n+1}{4} + \frac{n-1}{2} = \frac{n^2-1}{4}$.

22. d

La un arbore plin cu n noduri avem $n = 2k - 1$, unde k este numărul de noduri frunză și $k - 1$ este numărul de noduri interne. k este o putere întreagă a lui 2: $k = 2^x$. Există un singur arbore care cel mai adânc. Acesta este un arbore degenerat care se formează urmând succesorii din stânga, pe calea de la rădăcină către frunză. Adâncimea acestuia este aceeași cu adâncimea arborelui inițial. Numărul de noduri frunză: $k = \frac{n+1}{2}$. Adâncime arbore: $log_2 k = log_2 \frac{n+1}{2}$.

23. b

Inițial se adaugă muchii doar la nivelul celor 3 componente conexe, până când acestea devin subgrafuri complete. Numărul minim de muchii pentru o componentă conexă cu n noduri este $n - 1$ (lanț care conține toate nodurile). Numărul maxim de muchii pentru o componentă conexă cu n noduri este $\frac{n(n-1)}{2}$ (graf complet). Numărul de muchii adăugate: $\frac{3(3-1)}{2} + \frac{4(4-1)}{2} + \frac{5(5-1)}{2} - 2 - 3 - 4 = 10$. Din acest punct, se pot adăuga muchii doar între componentele conexe. Pentru a avea două componente conexe la final, se pot adăuga muchii doar între 2 dintre cele 3 componente conexe existente. Pentru un maxim de muchii, se conectează componentele conexe cu cel mai mare număr de noduri: 4, respectiv, 5. Numărul de muchii adăugate: $4 * 5 = 20$. Număr total de muchii: $10 + 20 = 30$.

24. b

Elementele cozii dupa inserare și extragerea a două valori vor fi: 25, 16, 9, 4, 1. În vârful stivei va fi 36. Diferența este 11.

25. a