Rezolvare testele 9 si 8 propuse pentru Bac 2021

Testul 9

Subiectul I

1.
   - Rezolvare:
     - a -> Daca luam x = 25 obtinem: (24)*10+5 = 245
     - b -> Daca luam x = 25 obtinem: 2+4+5 = 11
     - c -> Daca luam x = 25 obtinem: 54*10+2 = 542
     - d -> Daca luam x = 25 obtinem: 6 * 10 + 5 = 65
   - Raspuns: a
2.
   - Rezolvare:

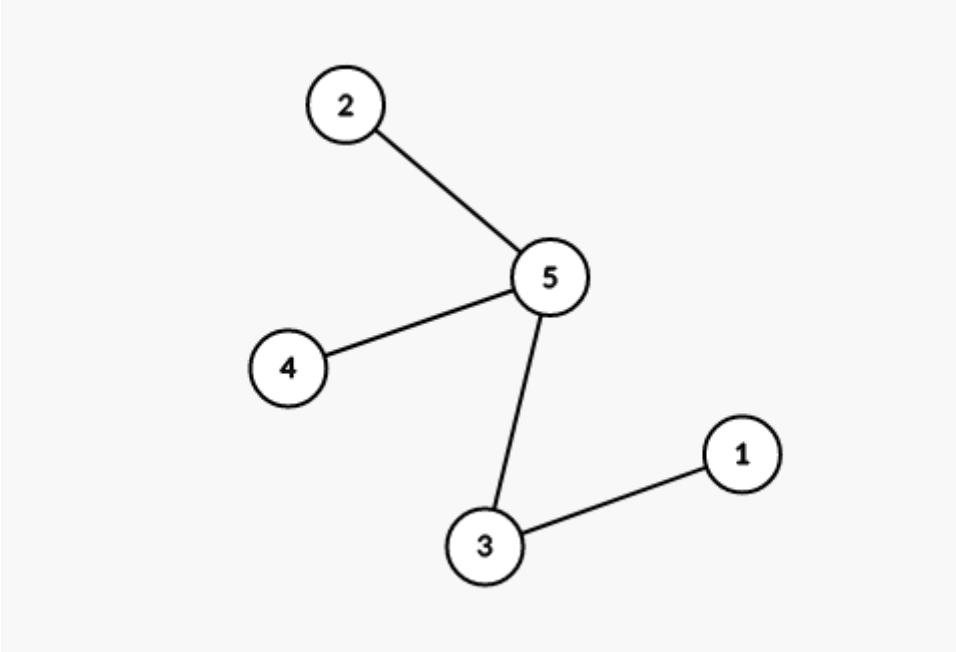
```
    Notam
    {salcie, carpen, larice, fag, ulm}
    {  0       1       2      3    4 }

    Primele 4 solutii sunt
    (salcie, carpen, larice, fag, ulm)
    (salcie, carpen, larice, ulm, fag)
    (salcie, carpen, fag, larice, ulm)
    (salcie, carpen, fag, ulm, larice)

    Adica:
    (0 1 2 3 4)
    (0 1 2 4 3)
    (0 1 3 2 4)
    (0 1 3 4 2)

    Deoarece sunt multe variante de generat, putem pleca de la
ultima varianta generata anume

    Ultima: -> 4 3 2 1 0
    Penultima: -> 4 3 2 0 1
    Antepenultima: -> 4 3 1 2 0 (ulm fag carpen  larice salcie)
- Raspuns corect: `c`
```

3.
   - Rezolvare:

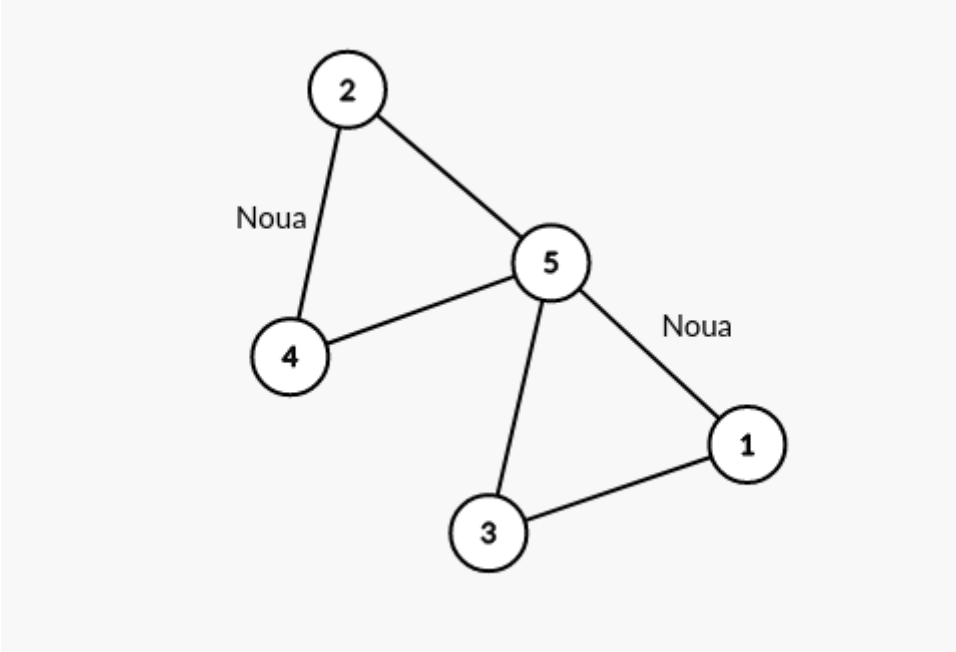
```
    Eliminam a,b,c din punct de vedere al sintaxei.
```

   - Raspuns corect: d
4.
   - Rezolvare:

        - Dupa ce desenam arborii, observam ca pentru nodurile 3 si 5 avem numar maxim de noduri pentru nivelul 2 (3)
    - Raspuns corect: a
5. - Rezolvare:
        - Conform enuntului avem graful de mai jos:



        - Lanțul care conține numai muchii distincte este lanț simplu. Dacă muchiile unui lanț nu sunt distincte se numește lanț compus.
        - Definiție: Se numește ciclu un lanț simplu în care primul vârf este identic cu ultimul. Dacă toate vârfurile sunt distincte, mai puțin primul și ultimul, se numește ciclu elementar.
        - Tinand cont de teorie, daca adaugam muchiile si obtinem ciclul: `3,5,4,2,5,1,3` conform pozei de mai jos:



    - Raspuns corect: b

Subiectul II

1.
   - a

```
n = 3
i = 1
    j = 1
        scrie "+"
    j = 2
        scrie "+"
    j = 3
        scrie "+"
    scrie "@"
i = 2
    j = 2
        scrie "+"
    j = 3
        scrie "+"
i = 3
    j = 3
        scrie "+"
    scrie "@"
```

     - Programul afiseaza: "+++@+++@"
   - b

```
4, 5
```

   - c

```
# include <iostream>

using  namespace std;

int main() {
    int n;
    cin >> n;
    for (int i = 1; i <=n; i++) {
        for (int j = 1; j <= n; j++) {
            cout << "+";
        }

        if (i % 2 != 0) {
            cout << "@";
        }
    }
}
```

- d

```
citeşte n
(număr natural nenul)
i <- 1
┌cat timp i<=n execută
│┌pentru j<-i,n execută
│| scrie '+'
│└■
│┌dacă i%2≠0 atunci
│| scrie '@'
│└■
| i<- i+1
└■
```

2.
   - Rezolvare:
     - Aici trebuie mai intai sa dam valori pentru a observa cum merge functia
     - Incepem cu x = 1 si avem:

```
f(10, 1) =
    = 10/1 + f(9, 1) =
        = 9/1 + f(8, 1) =
            = 8/1 + f(7, 1) =
                = 7 + f(6, 1) =
                    = 6 + f(5, 1) =
                        = 5 + f(4, 1) =
                            = 4 + f(3, 1) =
                                = 3 + f(2, 1) =
                                    = 2 + f(1, 1) =
                                        = 1
                                    = 3
                                = 6
                            = 10
                        = 15
                    = 21
                = 28
            = 36
        = 45
    = 55
```

     - Deci pentru 1 am obtinut 55. Acum crestem putin si calculam f(10, 3)

```
f(10, 3) =
= 10/3 + f(7, 3) =
    = 7/3 + f(4, 3) =
        = 4/3 + f(1, 3) =
            = 3/1 + f(1, 2) =
                = 2/1 + f (1, 1) =
```

```
                        = 1
                    = 3
                = 6
            = 7
        = 9
    = 12
```

- Si observam ca pentru x = 3 nu ajungem, mai incercam cu x = 2
- Calculam f(10, 2)

```
f(10,2) =
    = 10/2+ f(8, 2) =
        = 8/2 + f(6, 2) =
            = 6/2 + f(4, 2) =
                = 4/2 + f(2, 2) =
                    = 1
                = 3
            = 6
        = 4+ 6 = 10
    = 15
```

- Pana acum, ce putem observa este ca cu cat e mai mare diferenta intre parametri, cu atat is sansele mai mari, deci mai ramane sa incercam cu x = 9;
- Calculam f(10, 9)

```
f(10, 9) =
    = 10/9 + f(1, 9)
        = 9/1 + f(1, 8)
            = 8/1 + f(1, 7)
                = 7/1 + f(1, 6)
                    = 6/1 + f(1, 5)
                        = 5/1 + f(1, 4)
                            = 4/1 + f(1, 3)
                                = 3/1 + f(1, 2)
                                    = 2/1 + f(1, 1)
                                        = 1
                                    = 3
                                = 6
                            = 10
                        = 15
                    = 21
                = 28
            = 36
        = 45
    = 46
```

- Raspuns corect: `1, 9`

- Din pacate aici, la bac, o sa trebuiasca sa faci pentru fiecare dintre cazuri, sa nu risti sa gresesti.

3. Rezolvare:

```
    # include <iostream>

    using  namespace std;
    int f(int a, int b);

    int main() {
        int matrice[4][5];
        for(int i = 0; i < 4; i++) {
            for(int j = 0; j < 5; j++) {
                    matrice[i][j]= (i+j) % 3;
            }
        }

        for(int i = 0; i < 4; i++) {
            for(int j = 0; j<5; j++) {
                    cout << matrice[i][j] <<" ";
            }
            cout << endl;
        }
    }
    // Pentru bac, tot ce trebuia sa scrii in acea linie punctata
este
    // matrice[i][j]= (i+j) % 3;
    // Se observa usor cum la fiecare pas, valoarea este egala cu
restul impartirii la 3 a sumei indicelui liniei si coloanei.
```

Subiectul III

1. Rezolvare

```
    # include <iostream>
    using  namespace std;
    void divizor(int a, int b, int k, int &nr);

    int main() {
        int a = 3, b = 50, k = 4, nr;
        divizor(a, b, k, nr);
        cout << nr;
    }

    void divizor(int a, int b, int k, int &nr) {
        int contor = 0;
        for (int i = a; i<=b; i++) {
            if (i % k == 0 && i/10 == k) {
                contor++;
            }
```

```
        }
        nr = contor;
    }
```

2. ◦ Rezolvare

```
    #include <iostream>
    #include <cstring>

    using  namespace std;

    int main() {
        char text[101];
        cin.getline(text, 101);
        char* cuvant = strtok(text, " ");
        int exista = 0;
        while (cuvant != NULL) {
            int valid = 1;
            char ultimaVocalaCitita = 0;
            for(int i = 0; i < strlen(cuvant); i++) {
                if (strchr("aeiou", cuvant[i])) {
                    // daca e prima vocala citita, o salvam
                    if (ultimaVocalaCitita == 0) {
                        ultimaVocalaCitita = cuvant[i];
                        // altfel, verificam sa vedem daca e
diferita de
                        // ultimca vocala citita
                    } else if (ultimaVocalaCitita != cuvant[i]) {
                        valid = 0;
                        break;
                    }
                }
            }
            if (valid == 1) {
                exista = 1;
                cout << cuvant << endl;
            }
            cuvant = strtok(NULL, " ");
        }

        if (!exista) {
            cout << "nu exista";
        }
    }
```

3. ◦ a

```
    O sa implementam un algoritm care va calcula frecventa
tuturor cifrelor ale numerelor citite. Practic citim numar cu
numar si pentru fiecare, actualizam vectorul de frecventa conform
cifrelor fiecarui numar. Dupa care, la final, parcurgem vectorul
de frecventa de la capat si pentru fiecare cifra, o afisam de
atatea ori cat este valoarea frecventei sale. Algoritmul este
eficient din punct de vedere al timpului de executie deoarece
fisierul este parcurs o singura data si in acelasi timp, este
eficient din punct de vedere al memoriei, deoarece din totalul de
10^5 numere cat pot exista in fisier, noi in memorie, vom avea
doar 10, anume cele din vectorul de frecventa.
```

- b

```
#include <iostream>
#include <fstream>

using  namespace std;

int main() {
    ifstream fin("numere.txt");
    int frecventa[10] = {0};
    int numar;
    while(fin >> numar){
        while(numar) {
            int ultimaCifra = numar %10;
            frecventa[ultimaCifra]++;
            numar /= 10;
        }
    }

    for(int i = 9; i >=0; i--) {
        if (frecventa[i] != 0) {
            for(int j = 0; j < frecventa[i]; j++) {
                cout << i;
            }
        }
    }

    fin.close();
}
```

Testul 8

Subiectul I

1.
   - Rezolvare:
     - Evaluam fiecare optiune atat cu un numar cu 2 cifre cat si cu 3 cifre
       - a -> Nu exista numar cu 2 cifre care sa dea 0 la x/10

        - b -> Aici obtinem 0 si pentru numere cu mai mult de 2 cifre (e.g 3000)
        - c -> Aici obtinem 0 doar pentru numere de 2 cifre sau mai putin.
        - d -> Aici putem sa obtinem 0 si pentru numere mai mari de 2 cifre (e.g 200)
    - Raspuns corect: `c`
2. - Rezolvare:
        - Calculam pentru 4 mai intai

```
f(4) =
    = g(4+f(3))
        = g(3+f(2))
            = g(2+f(1))
                = g(1+f(0))
                = g(2) = 2
            = g(2+2) = g(4) = 4
        = g(3+4) = g(7) = 7
    = g(4+7) = g(11) = 1 + 1 = 2
```

        - Acum sa incercam si pentru 6

```
f(6) =
    = g(6 + f(5))
        = g(5 + f(4)) // stim ca f(4) = 2
        = g(5+2) = g(7)
    = g(6+7) = g(13) = 4
```

        - Vedem ca a este invalid. trecem la b si incepem cu 7
        - Calculam f(7)

```
f(7) =
    = g(7 + f(6)) // stim ca f(6) = 4
    = g(7+4) = g(11) = 2
```

        - Calculam f(9)

```
f(9) =
    =g(9 + f(8))
        = g(8 + f(7)) // stim ca f(7) = 2
        = g(10) = 1
    =g(9+1) = g(10) = 1
```

        - Am eliminat si a si b. verificam c
        - Calculam f(1)

```
f(1)
    = g(1 + f(0)) = g(2) = 2
```

- Stiin ca f(1) = f(4) = f(7) putem deduce clar ca d este varianta corecta

- Raspuns corect: d

3. - Rezolvare:

```
Primele 3 solutii sunt:
[cămașă, cravată, pantaloni, sacou, șosete, pantofi],
[cămașă, cravată, pantaloni, șosete, pantofi, sacou],
[cămașă, cravată, pantaloni, șosete, sacou, pantofi]

Adica:
[0, 1, 2, 4, 5 3],
[0, 1, 2, 5, 3, 4],
[0, 1, 2, 5, 4, 3]

Practim din cele 6 pozitii, conform cerintei avem asa:

X X _ _ _ _  Unde pe primele 2 pozitii vom avea sigur camasa
si cravata.
Asta inseamna ca ne simplificam putin ce avem de generat
ramanand doar cu
[pantaloni, pantofi, sacou, șosete]
Si acum sa vedem ce generam
[sosete, pantaloni, pantofi, sacou]
[sosete, pantaloni, sacou, pantofi]
[pantaloni, sosete, pantofi, sacou]
[pantaloni, sosete, sacou, pantofi]
[pantaloni, sacou, sosete, pantofi]
[sosete, sacou, pantaloni, pantofi]
[sacou, pantaloni, sosete, pantofi]
[sacou, sosete, pantaloni, pantofi]
```

- Raspuns corect: b

4. - Rezolvare:
   - Conform enuntului avem:

```
[1,2,3,4,5,6,7,8,9]
[5,3,0,1,3,3,8,3,1]

Adica:
3 Radacina si parinte pentru: 2, 5, 6, 8
Deci vedem ca un nod frate cu 6 este 5
```

- Raspuns corect: c

5. - Rezolvare:

```
    - 202 elemente nenule inseamna ca avem 202 de elemente = 1
    - Graf neorientat, stim ca daca avem 1 pentru matrice[i][j]
avem 1 si pt matrice[j][i]
    - Deci practic, inseamna ca avem un total de 101 muchii si
care este numarul minim de componente conexe pe care le putem
face cu aceste 101 muchii
    - Daca avem 101 muchiii, putem lega maxim 102 noduri (poti sa
il vezi ca pe un arbore, unde daca ai n noduri, ai n-1 muchii)
    - Deci inseamna ca avem 2021 -102 = 1919 de noduri izolate
    - Deci in total avem 1919 + componenta unde am pus cele 102
noduri deci avem 1920
```

- Raspuns corect: d

Subiectul II

1.
   - a

```
    n = 10
    x = 0
    i = 1
        x = 0 + 2 * 2 = 4
    i = 2
        x = 4 + 4 = 8
    i = 3
        x = 8 + 4 * 4 = 24
    i = 4
        x = 24 + 16 = 40
    i = 5
        x = 40 + 36 = 76
    i = 6
        x = 76 + 36 = 112
    i = 7
        x = 112 + 8 * 8 = 112 + 64 = 176
    i = 8
        x = 176 + 64 = 240
    i = 9
        x = 240 + 100 = 340
    i = 10
        x = 340 + 100 = 440
```

     - Se va afisa 440
   - b

```
    - Putem observa ca la suma se adauga de 2 ori patratul
fiecarui numar par de pana la limita
        - Practic pana la 10 am adaugat:
            - 2 * 4 = 8 // patratul lui 2
```

```
            - 2 * 16 = 32 // patratul lui 4
            - 2 * 36 = 72 // patratul lui 6
            - 2 * 64 = 128// patratul lui 8
            - 2 * 100 = 200 // patratul lui 10
            - Care in total da 440
        - Deci daca mergem pana la 6 obtinem 112 si este cea mai
mica valoare pe care o putem obtine din intervalul cerut.
        - Acum trebuie sa vedem cat putem citii maximum pentru a
ne apropia de 999 dar a nu-l depasi:
             - 2 * 4 = 8 // patratul lui 2
            - 2 * 16 = 32 // patratul lui 4
            - 2 * 36 = 72 // patratul lui 6
            - 2 * 64 = 128// patratul lui 8
            - 2 * 100 = 200 // patratul lui 10
            - 2 * 144 = 288 // patratul lui 12
        - Deci daca mergem pana la 12 avem: 728
        - Daca mergem pana la 13 avem 924 si vedem ca aici ne
oprim
```

- Raspuns corect: 6, 13

- c

```c
#include <iostream>

using  namespace std;

int main() {
    int n;
    cin >> n;
    int x = 0;
    for (int i = 1; i <= n; i++) {
        if (i % 2 == 0) {
            x = x + i * i;
        } else {
            x = x + (i+1) * (i+1);
        }
    }
    cout << x;
}
```

- d

```
citește n (număr natural) x□0
i<- 1
┌cat timp i<=n execută
│ ┌dacă i%2=0 atunci
│ │ x<-x+i*i
│ │altfel
```

```
      | | x<-x+(i+1)*(i+1)
      | └■
      | i <- i+1
      └■
      scrie x
```

2. ○ Rezolvare:

```
    struct elev_clasa {
        double sem1;
        double sem2;
    };

    struct clasa {
        int numar;
        elev_clasa elev[41];
    }p;
```

3. ○ Rezolvare

```
    Mai intai se afiseaza lungimea stringului "voalata" adica:
"7"
    Dupa aceea, de la 0 pana la 6 inclusiv, daca intalnim o
vocala, o stergem din sir si ne mutam la caracterul urmator Insa
daca nu e vocala, sarim 2 caractere:
    Astfel:
    i = 0 -> voalata nu avem vocale, deci sarim la i = 2
    i = 2 -> avem vocala 'a' rezulta s = "volata"
    i = 3 -> avem vocala 'a' rezulta s = "volta"
    i = 4-> avem vocala 'a' rezulta s = "volt"
```

- Deci programul afiseaza "7volt"

Subiectul III

1. ○ Rezolvare:

```
    #include <iostream>

    using  namespace std;

    void nrfp(int n, int &m);

    int main() {

        int n = 100, m;
        nrfp(n, m);
```

```
        cout << m;
        return 0;
    }

    void nrfp(int n, int &m) {
        int max = 2;
        int numarTermeniMax = 1;
        for(int i = 2; i<= n; i++) {
            int numarTermeniCurent = 0;
            for(int j = 2; j*j<= i; j++) {
                // verificam daca avem un divizor si daca e
adevarat, atunci
                // verificam si daca acesta este prim
                if (i % j == 0) {
                    // j este divizor, acum vedem daca e si prim
                    int estePrim = 1;
                    for(int k = 2; k*k <=j; k++) {
                        if (j % k == 0) {
                            estePrim = 0;
                            break;
                        }
                    }
                    if (estePrim) {
                        numarTermeniCurent++;
                    }
                }
            }
            if (numarTermeniCurent >= numarTermeniMax) {
                numarTermeniMax = numarTermeniCurent;
                max = i;
            }
        }

        m = max;
    }
```

- Nota:

```
Cu siguranta exista o metoda mai eficienta insa am mers pe
solutia cea mai simpla, anume pentru fiecare numar, ii aflam
divizorii si in acelasi timp vedem cati dintre divizori sunt
primi.
```

2. - Rezolvare:

```
    #include <iostream>

    using  namespace std;
```

```
int main() {
    int n;
    cin >> n;
    int matrice[n][n];
    int suma = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> matrice[i][j];
        }
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (j > i && j > (n-1-i) && j != n-1) {
                suma += matrice[i][j];
            }
        }
    }

    cout << suma;
    return 0;
}
```

3.
   - Rezolvare:
     - a

```
    Vom scrie un algoritm care va initializa doua variabile
in care vom tine primulImpar de afisat si cel de al doilea.
Valoarea acestora, de inceput, va fi -1 pentru a sti daca
ele au fost sau nu populate.
    Dupa accea, in timp ce parcurgem fisierul, atunci cand
gasim un numar impar, initializam pe rand cele doua
variabile, atunci cand ambele au fost initializate, la
urmatorul numar impar citit, mutam ce aveam in a doua
variabila in prima variabila si a doua variabila va contine
numarul impar curent. Daca la final, vedem ca cel putin una
din cele 2 variabile nu a fost initializata, afisam "nu
exista".
    Acest algoritm este eficient din punct de vedere al
timpului de executie deoarece fisierul este citit o singura
data, si totodata este eficient din punct de vedere al
memoriei deoarece noi tinem in  memorie cel mult 2 numere
din totalul de 1_000_000 de numere posibile in fisier.
```

     - b

```
#include <iostream>
#include <fstream>
```

```cpp
using  namespace std;

int main() {
    ifstream fin("bac.txt");
    int numar;
    int primulImpar = -1;
    int alDoileaImpar = -1;
    while (fin >> numar) {
        if (numar % 2 != 0) {
            if (primulImpar == -1) {
                primulImpar = numar;
            } else if (alDoileaImpar == -1) {
                alDoileaImpar = numar;
            } else {
                primulImpar = alDoileaImpar;
                alDoileaImpar = numar;
            }
        }
    }

    if (primulImpar == -1 || alDoileaImpar == -1) {
        cout << "nu exista";
    } else {
        cout << primulImpar << " " << alDoileaImpar;
    }

    return 0;
}
```