Rezolvare varianta 7 din cartea lui Vlad

Subiectul I

1.
   - Rezolvare
     - Din enunt stim ca avem 2 intervale care nu se intersecteaza (atentia la intervale)
       - De exemplu a = 1 si b = 3, si c= 5 si d= 8
     - Acum haideti sa luam fiecare optiune si sa gasim pe cea corecta
       - a -> Aici putem sa obtinem 1 (adevarat) si daca intersectia intervalelor nu este vida deci optiunea nu este corecta
         - de exemplu a = 2, b = 3, c = 1, d = 6
       - b -> Din nou, si aici putem sa obtinem 1 (adevarat) si daca intersectia intervalelor nu este vida deci optiunea nu este corecta
         - de exemplu a = 3, b = 6, c = 3, d = 8
       - c -> Acesta optiune cade deoarece stim ca numerele au proprietatea ca a <=b si c<=d. Si din ce vedem in optiune, daca a>d, rezulta implicit ca a >=c deci prin urmare b nu poate sa fie mai mic decat c, deoarece b este cel putin egal cu a
       - d -> Optiunea este adevarata deoarece conform proprietatii si conditiei din enunt, obtinem 1 (adevarat) doar daca se respecta conditia din enunt si proprietatea celor 4 numere
         - de exemplu a = 2, b = 4, c = 4 d = 7 si aici vedem ca (a, b) nu se intersecteaza cu [c,d]
   - Raspuns corect: d
2.
   - Rezolvare:

```
scrie(2021, 2023) =
    scrie (2022, 2023) =
        = scrie (2023, 2023)
            = afiseaza: "2023 2023"
```

     - Raspuns corect: c
3.
   - Rezolvare:
     - Conform enuntului stim ca avem combinatii de forma:
       - `_ _ _ _ _ Lebada`
       - Mai stim ca leul tigrul si ursul nu pot sa fie vecini cu cerbul Asa ca o sa avem doua situatii:
         1. punem cerbul pe prima pozitie si lebada pe ultima. Asta automat inseamna ca o sa punem papagalul pe a doua pozitie si o sa restrangem combinatiile la forma:
            - `Cerb Papagal _ _ _ Lebada`
         2. Punem cerbul intre papagal si lebada asta insemnand ca avem combinatii de forma:
            - `_ _ _ Papagal Cerb Lebada`

        - Acum stim ca cele 3 locuri libere in ambele situatii pot fi alocate catre leu, tigru si urs astfel:
            - Leu Tigru Urs
            - Leu Urs Tigru
            - Tigru Leu Urs
            - Tigru Urs Leu
            - Urs Leu Tigru
            - Urs Tigru Leu
        - In concluzie avem 6 posibile combinatii pentru fiecare situatie, deci in total `12`
    - Raspuns corect `d`
4.
    - Rezolvare:
        - Conform enuntului stim ca avem 2022 de varfuri si 1000 de muchii.
        - 1000 de muchii inseamna ca automat avem 1000 de valori `1` in matrice. Faptul ca este neorientat, dubleaza acest numar la 2000 deoarece daca, sa zicem varful 1 este conectat cu varful 3, in matrice o sa avem `1` atat pentru muchia `(1,3)` cat si pentru muchia `(3,1)`
    - Raspuns corect: `c`
5.
    - Rezolvare:
        - a -> Fals deoarece arborele este un graf conex aciclic
        - b-> Fals deoarece eliminand o muchie, cel putin un nod va deveni izolat, deci conexitatea va fi eliminata
        - c -> Adevarat, conform teoriei, arborele este un graf conex cu n-1 muchii [https://www.pbinfo.ro/articole/5982/arbori-cu-radacina]
        - d -> Fals, acelasi lucru ca la a
    - Raspuns corect `c`

Subiectul II

1.
    - a

```
x = 20, y = 80, z = 18
x > y fals
 x <= y adevarat
    x % z == 0 fals
    x = 21
x <= y adevarat
    x % z == 0 fals
    x = 22
... mergem pana la 35
x <= y adevarat
    x % z == 0 fals
    x = 36
x <= y adevarat
    x % z == 0 adevarat
        afisam x => afisam 36
    x = 37
... mergem pana la 53
x <= y adevarat
    x % z == 0 fals
```

```
        x = 54
    x <= y adevarat
        x % z == 0 adevarat
            afisam x => afisam 54
        x = 55
    ... mergem pana la 71
    x <= y adevarat
        x % z == 0 fals
        x = 72
    x <= y adevarat
        x % z == 0 adevarat
            afisam x => afisam 72
        x = 73
    ... sfarsit
```

- program afiseaza toti multipli lui z in intervalul [x, y]. Iar programul de mai sus va afisa: `36, 54, 72,`

- b

```
    Daca x = 100, y = 200, pentru a se afisa doar valoarea 150,
cea mai mare valoare citita pentru z trebuie sa fie egala cu 150
```

- Nota: rezolvarea in carte alege 75 insa nu inteleg de unde scoate conditia care zice ca z sa nu faca parte din intervalul [x, y]

- c

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

    int main() {
        int x,y,z;
        cin >> x >> y >> z;
        if (x > y) {
            int aux = x;
            x = y;
            y = aux;
        }
        while (x <= y) {
            if (x % z == 0) {
                cout << x << ", ";
            }
            x = x+1;
        }
        return 0;
    }
```

- d

```
daca x > y atunci
    aux <- x
    x <- y
    y <- aux
pentru i<-x,y executa
    daca i % z = 0 scrie i, " "
```

2. ◦ Rezolvare:

```
float a = x.y * x.y
```

3. ◦ Rezolvare:
   - Teorie graf partial: https://www.ezinfo.ro/XI/gorientate/partial.html
   - Din matricea de adiacenta observam ca avem 8 arce
   - Rezolvare:

```
    Numărul maxim de grafuri parțiale ale unui graf orientat
cu m arce este: 2^m.
    - In cazul nostru, putem avea 2^8 => 256
```

Subiectul III

1. ◦ Rezolvare:

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

void nrmax(int &n);

int main() {
    int n = 1372435;
    nrmax(n);
    cout << n;
    return 0;
}

void nrmax(int &n) {
    int contorPare = 0;
    int contorImpare = 0;
    while (n) {
        int ultimaCifra = n % 10;
        if (ultimaCifra % 2 == 0) {
            contorPare++;
        } else {
            contorImpare++;
```

```
            }

            n = n/10;
        }

        if (contorImpare > contorPare) {
            n = contorImpare;
        } else {
            n = contorPare;
        }
    }
```

2.
   - Nota:
     - problema putea fi rezolvata mult mai simplu folosind functia strtok insa am vrut sa exemplific cum am putea extrage cuvintele dintr-un sir
     - o alta motivatie pentru complicaciunea pe care am facut-o este ca am tratat cazul in care avem mai multe spatii consecutive, lucru care s-ar fi pierdut daca am fi folosit functia strtok
       - acum, ce e drept, enuntul nu spune daca ar putea fi sau nu mai multe caractere de tip spatiu, consecutive.
   - Rezolvare:

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

    int main() {
        ofstream fout("bac.txt");
        char text[101];
        cin.getline(text, 101);
        int lungimeText = strlen(text);
        int pozitieInceputCuvant=-1;
        for (int i = 0; i < lungimeText;i++) {
            if ((text[i] == ' ' || i == lungimeText-1) &&
pozitieInceputCuvant != -1) {
                // am gasit un cuvant, il extragem si efectuam
modificarile
                int lungimeCuvant;
                if (i == lungimeText-1) { // daca suntem la
finalul sirului
                    lungimeCuvant = i - pozitieInceputCuvant + 1;
                } else {
                    lungimeCuvant = i - pozitieInceputCuvant;
                }
                char cuvant[lungimeCuvant];
                strncpy(cuvant, text+pozitieInceputCuvant,
lungimeCuvant);
                cuvant[lungimeCuvant] = '\0';
                pozitieInceputCuvant = -1;
```

```
                char oglindit[lungimeCuvant];
                for (int j = strlen(cuvant)-1, k = 0; j >= 0; j-
-, k++) {
                    oglindit[k] = cuvant[j];
                }
                oglindit[strlen(cuvant)] = '\0';
                if (strcmp(cuvant, oglindit) != 0) {
                    fout << oglindit << " ";
                } else {
                    // tratam cazul in care avem mai mult spatii
consecutive.
                    fout<< cuvant << " ";
                }
            } else if (text[i] == ' ') {
                fout << " ";
            }
            else if (text[i] != ' ' && pozitieInceputCuvant ==
-1) {
                pozitieInceputCuvant = i;
            }
        }
        return 0;
    }
```

3. ◦ Rezolvare:
   - a

```
    Algoritmul de la punctul b va citii cate un numar din
fiecare fisier si il va afisa pe cel mai mare dintre ele. In
acelasi timp, vom salva ultimul numar afisat pentru a avita
afisarea numerelor care sunt duplicate. La final, ne
asiguram ca afisam ce a mai ramas din fiecare fisier.
Algoritmul este eficient din punct de vedere al timpului de
executie deoarece se face o singura parcurgere a fisierelor.
Algoritmul este eficient si din punct de vedere al memoriei
utilizate deoarece nu se folosesc alte structuri pentru a
stoca numerele din cele 2 fisiere.
```

   - b

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

    int main() {
        ifstream fin1("date1.txt");
        ifstream fin2("date2.txt");
```

```
        int n, m;
        fin1 >> n;
        fin2 >> m;

        int numarDate1, numarDate2, ultimNumarAfisat = 0;
        fin1 >> numarDate1;
        fin2 >> numarDate2;
        int i = 1, j = 1;
        while (i <= n && j <= m) {
        if (numarDate1 > numarDate2) {
            if (numarDate1 != ultimNumarAfisat) {
                cout << numarDate1 << " ";
            }
            ultimNumarAfisat = numarDate1;
            fin1 >> numarDate1;
            i++;
        } else {
            if (numarDate2 != ultimNumarAfisat) {
                cout << numarDate2 << " ";
            }
            ultimNumarAfisat = numarDate2;
            fin2 >> numarDate2;
            j++;
        }
        }

        if (numarDate1 != ultimNumarAfisat) {
            cout << numarDate1 << " ";
        }
        while (fin1 >> numarDate1) {
            cout << numarDate1 << " ";
        }

        if (numarDate2 != ultimNumarAfisat) {
            cout << numarDate2 << " ";
        }
        while (fin2 >> numarDate2) {
            cout << numarDate2 << " ";
        }


        return 0;
    }
```

- c -> varianta care nu e eficienta din punct de vedere al memoriei utilizate

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

    int main() {
        ifstream fin1("date1.txt");
```

```
        ifstream fin2("date2.txt");
        int n, m;
        fin1 >> n;
        fin2 >> m;
        int aparitii[100] = {0};

        for (int i = 0; i < n; i++) {
            int numar;
            fin1 >> numar;
            aparitii[numar]++;
        }

        for (int i = 0; i < m; i++) {
            int numar;
            fin2 >> numar;
            aparitii[numar]++;
        }

        for (int i = 99; i >= 0; i--) {
            if (aparitii[i] > 0) {
                cout << i << " ";
            }
        }

        return 0;
    }
```