Varianta culeasa din culegerea bacalaureat la informatica - Teste Rezolvate (2023) V2

Subiectul I

1. d
2. c
3. d
4. d
5. d -> Atentie la faptul ca este vorba de un graf conex deci nu putem avea un nod izolat!

Subiectul II

1. Programul
    - a. 4351. Dupa cum se observa, se compara ultima cifra din fiecare numar. Se va elimina din numarul care va contine cifra cea mai mare a unitatilor.
    - b. Pentru variabila `a` poate fi citit orice numar natural de 4 cifre care are ultima cifra egala cu 0. Astfel avem:
        - 9 valori posibile pentru prima cifra
        - 10 valori posibile pentru a doua cifra
        - 10 valori posibile pentru a treia cifra
        - In total avem 9x10x10 = 900 de numere care pot fi citita pentru variabila `a`
    - c. Vom inlocui cu structura `executa .. cat timp`:

```
citeste a,b (numere naturale)
nr <- 0
daca a > 0 si b > 0 atunci
    executa
        daca a % 10 > b % 10
            atunci c<- a%10; a<- [a/10]
            altfel c<- b%10; b<- [b/10]
        nr <- nr * 10 + c
    cat timp a > 0 si b > 0
```

    - d

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

int main()
{
    int a, b, nr = 0;
    cin >> a >> b;
    while(a > 0 && b > 0) {
```

```
            int c;
            if (a%10 > b%10) {
                c = a%10;
                a = a/10;
            } else {
                c = b%10;
                b = b/10;
            }
            nr = nr * 10 + c;
        }
        cout << nr;

        return 0;
    }
```

2. `abs(C.O.y) <= C.R`

3. Matricea rezultata este:

```
6   5 4 3 2 1
7   6 5 4 3 2
8   7 6 5 4 3
9   8 7 6 5 4
10  9 8 7 6 5
11  10 9 8 7 6
```

- Raspuns corect: `1 3 5 7 9 11`

Subiectul III

1. Solutie:

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

void sub(int n, int v[]);
int main()
{
    int v[] = {35, 1, 52, 98, 1, 98, 51, 11, 98, 65};
    int n = 10;
    sub(n, v);

    for (int i = 0; i < n; i++) {
        cout << v[i] << " ";
    }
    return 0;
}

void sub(int n, int v[]) {
```

```
        int min = v[0];
        int pozitieMin = 0;
        int max = v[0];
        int pozitieMax = 0;
        for(int i = 1; i < n; i++) {
            if (v[i] < min) {
                min = v[i];
                pozitieMin = i;
            }
            if (v[i] >= max) {
                max = v[i];
                pozitieMax = i;
            }
        }

        if (max == min){
            return;
        } else {
            int aux = v[pozitieMax];
            v[pozitieMax] = v[pozitieMin];
            v[pozitieMin] = aux;
        }
    }
```

2. Solutie:

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

    int main()
    {
        char s[255];
        cin >> s;
        char rezultat[255]="";
        char * cuvant = strtok(s, "*");
        char sufix[255];
        strcpy(sufix, cuvant);
        while(cuvant != NULL) {
            int esteSufix = 1;
            if (strlen(cuvant) >= strlen(sufix)){
                int lungimeCuvant = strlen(cuvant);
                int lungimeSufix = strlen(sufix);
                for (int i = lungimeCuvant - 1, j = lungimeSufix-1; i >= 0,
j>=0; i--, j--) {
                    if (cuvant[i] != sufix[j]) {
                        esteSufix = 0;
                        break;
                    }
                }
            } else {
                esteSufix = 0;
```

```
            }
            if (!esteSufix) {
                strcat(rezultat, cuvant);
            }
            strcat(rezultat, "*");
            cuvant = strtok(NULL, "*");
        }

        cout << rezultat;
        return 0;
    }
```

3. Solutie:

- a. Explicatie eficienta:
    - Solutia oferita este eficienta din punct de vedere al timpului deoarece parcurgem fisierul o singura data. Totodata, programul este eficient din punct de vedere al memoriei deoarece la orice pas, tinem minte doar 2 numere din fisier si nu se folosesc alte structuri de date.
- b. Cod c++:

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

    int sumaCifrelor(int n);

    int main()
    {
        ifstream fin("bac.in");
        int primulNumar, numarCurent, lungimeMaxima = 0, lungimeCurenta = 1;
        fin >> primulNumar;
        while (fin >> numarCurent) {
            if (sumaCifrelor(primulNumar) == sumaCifrelor(numarCurent)) {
                lungimeCurenta++;
            } else {
                primulNumar = numarCurent;
                if (lungimeMaxima < lungimeCurenta) {
                    lungimeMaxima = lungimeCurenta;
                }
                lungimeCurenta = 1;
            }
        }
        cout << lungimeMaxima;
        fin.close();
        return 0;
    }

    int sumaCifrelor(int n) {
        int sum = 0;
        while (n > 0) {
```

```
            sum += n%10;
            n/=10;
        }
        return sum;
    }
```