Varianta 6

Subiectul I

1. a (ATENTIE!!! => operatorii aritmetici +,-,*,/) au prioritate mai mare decat operatorii relationali
2. b
3. d (un graf este complet dacă numărul total de muchii este n(n-1)/2)
4. b (Se genereaza numerele: 1010, 1012, 1014, 1030, 1032, 1034, etc, al 4-lea numar fiind 1030)
5. b

Subiectul II

1. a. Se vor afisa numerele: `34, 100, 50` b. `json Algoritmul afiseaza maximul dintr-o secventa. Un alt exemplu ar putea fi: 21 4 5 6 0 7 8 21 8 8 0 11 15 17 21 0 0` c. `json citeste a,b (numere naturale nenule) m a ┌dacă a>0 sau b>0 atunci | ┌repeta | | a b | | citeste b | | ┌dacă a>m atunci | | |m a | | └■ | | ┌dacă b=0 și a>0 atunci | | | scrie m | | | m 0 | | └■ | └ până când a<=0 și b<=0 └■` d. ```c++ #include

```
using namespace std;
int main(){
    int a, b, m;
    cin >> a >> b;
    m = a;
    while (a > 0 || b > 0) {
        a = b;
        cin >> b;
        if (a > m) {
            m = a;
        }
        if (b == 0 && a > 0) {
            cout << m;
            m = 0;
        }
    }
}

```
```

2. Solutie:

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

struct data
```
```

```
    {
        int zi,luna,an;
    };
    struct elev
    {
        char nume[20];
        data datan;
    };
    elev m;

    int main() {

        cout << "Introduceti numele elevului (max 20 caractere): ";
        cin.getline(m.nume, 20);
        cout << "Introduceti data nasterii\n";
        cout << "Ziua: ";
        cin >> m.datan.zi;
        cout << "Luna: ";
        cin >> m.datan.luna;
        cout << "Anul: ";
        cin >> m.datan.an;

        if (m.datan.an < 2000) {
            cout << m.nume;
        }  else {
            cout << m.datan.zi <<"/"<<m.datan.luna<<"/"<<m.datan.an;
        }
        return 0;
    }
```

3. Solutie: `bcLura`.

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

using namespace std;

int main() {
    char a[31]= "BacaLauReat";
    int i;
    for (i = 0; i < strlen(a); i++) {
        if (a[i] >= 'A' && a[i] <= 'Z') a[i] = a[i] + 32;
        else {
            char t[31];
            strcpy(t, a + i + 1);
            strcpy(a + i, t);
        }

    }
    cout << a;
}
```

```
/*
* i = 0 => B => b (bacaLauReat
* i = 1 => copiem in t => caLauReat; copiem in a => bcaLauReat
* i = 2 => copiem in t => Laureat; copiem in a => bcLaureat
* i = 3 => copiem in t => uReat; copiem in a => bcLuReat
* i= 4=> a devine => bcLureat
* i = 5 => copiem in t => at; copiem in a => bcLurat
* i = 6 => nu copiem nimic in t si copiem in a bcLura
*/
```

Subiectul III

1. Solutie

```
#include <iostream>

using namespace std;

int p(int n, int x[], int k);

int main() {
    int n = 10;
    int k = 3;
    int x [] = {1,1,7,5,5,1,3,3,1,9};
    cout << p(n, x, k);
}

int p(int n, int x[], int k) {
    int arePare = 0;
    int maxPareGasite = 0;
    int pozitieSecventa = 0;

    for (int i = 0; i < n; i++) {
        int maxPareLocal = 0;
        for (int j = i; j < (i+k) && j < n; j++) {
            if (x[j] % 2 == 0) {
                arePare = 1;
                maxPareLocal++;
            }
        }
        if (maxPareLocal >= maxPareGasite) {
            maxPareGasite = maxPareLocal;
            pozitieSecventa = i;
        }
    }
    if (arePare == 0) {
        return -1;
    } else {
        return pozitieSecventa;
    }
```

```
}
```

2. Solutie:

```
#include <iostream>

using namespace std;

int main() {
    int n, m;
    cin >> n >> m;

    int matrice[n][m];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> matrice[i][j];
        }
    }
    int primaLinie=-1, ultimaLinie = -1;
    for (int i = 0; i < n; i++){
        for (int j = 0; j < m; j++) {
            // verificam daca toate elementele liniei sunt nenule
            int areNenule = 0;
            for (int k = 0; k < m; k++) {
                if (matrice[i][k] == 0) {
                    areNenule = 1;
                    break;
                }
            }
            if (areNenule == 0) {
                if (primaLinie == -1) {
                    primaLinie = ultimaLinie = i;
                } else {
                    ultimaLinie = i;
                }
            }
        }
    }

    // Evitam cazul in care avem o singura linie cu elemente nenule
    if (primaLinie != ultimaLinie) {
    for (int k = 0; k < m; k++) {
        int aux = matrice[primaLinie][k];
        matrice[primaLinie][k] = matrice[ultimaLinie][k];
        matrice[ultimaLinie][k] = aux;
    }
    }

    // afisam rezultatul
    for (int i = 0; i < n; i++){
        for (int j = 0; j < m; j++){
```

```
            cout << matrice[i][j] << " ";
        }
        cout << endl;
    }


}
```

3.

- a: Solutie:

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

int main() {
    ifstream fin("bac.in");
    int n, m;
    fin >> n >> m;
    int index = 1;
    int numar;
    while (fin >> numar) {
        if (numar == 1) {
            cout << ceil((index * 1.0)/m) <<" " << index%m<<";";
        };
        index++;
    }
    fin.close();
    return 0;
}
```

- b: Programul este eficient din punct de vedere al timpului de executie deoarece se efectueaza o singura citire a fisierului si in acelasi timp este eficienta si din punct de vedere al memoriei utilizate deoarece nu se folosesc alte structuri de date pentru a memora numerele, deoarece la orice pas, tinem minte doar numarul curent si un index, pe baza caruia determinam secventa si pozitia.