Subiectul I

1. d

2. b

3. Toate numerele care indeplinesc conditia sunt:

   - 1789
   - 1798
   - 1879
   - 1897
   - 1978
   - 1987
   - 2689
   - 2698
   - 2869
   - 2896
   - 2968
   - 2986
   - 3589
   - RASPUNS CORECT -> `c`

4. Atentie:

   - Graf neorientat complet: graf in care toate varfurile sunt conectate intre ele
   - Un graf neorientat complet are `(n * (n-1))/2` muchii
   - In cazul nostru avem in total 45 de muchii
   - Pentru a fi conex, graful nostru trebuie sa aiba minimum n-1 muchii, rezulta ca putem elimina `45 - 9 = 36` de muchii
   - RASPUNS CORECT -> `c`

5. a.

Subiectul II

1.
   - a. Se afiseaza numarul 3
     - Atentie la faptul ca algoritmul numara cate numere prime sunt intre `a` si `b`
   - b. 22 (Se vor afla primele 4 numere prime dupa 10 si ne vom opri cu 1 inainte de al 5-lea numar prim adica inainte de 23)
   - c.

```
citeste a,b
nr <- 0
pentru i <- a,b executa
    d <- 0
    j <- 2
```

```
        repeta
            daca i % j = 0 atunci
                d <- j
            j <- j+1
        pana cand j > [i/2]
        daca d = 0 atunci
            nr <- nr+1
    scrie nr
```

- d.

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

int main()
{
    int a, b, nr = 0;
    cin >> a >> b;
    for (int i = a; i <= b; i++) {
        int d = 0;
        for (int j = 2; j < i/2; j++) {
            if (i % j == 0) {
                d = j;
            }
        }
        if (d == 0) {
            nr = nr+1;
        }
    }
    cout << nr;
    return 0;
}
```

2.

```
if(strstr(t, s) == s) {
    cout<<"DA";
} else {
    cout<<"NU";
}
```

3. Se va afisa valoarea 5.

Subiectul III

1. Solutie

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

```
void cifre(int a, int &b);

int main()
{
    int a = 2334157, b;
    cifre(a, b);

    cout << b;

    return 0;
}

void cifre(int a, int &b) {
    int pozitie = 0;
    int rezultat = 0;
    int increment = 1;
    while (a > 0) {
        int ultimaCifra = a % 10;
        if (pozitie % 2 != 0) {
            rezultat = ultimaCifra * increment + rezultat;
            increment = increment * 10;
        }

        pozitie++;
        a = a / 10;
    }
    b = rezultat;
}
```

2. Solutie:

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

int main()
{
    int n;
    cin >> n;
    int valoare = 1;
    int matrice[n][n];
    for (int i = 0; i < n; i++) {
        if (i % 2 == 0) {
            for (int j = 0; j < n; j++) {
                matrice[i][j] = valoare;
                valoare = valoare + 1;
            }
        } else {
            for (int j = n-1; j >= 0; j--) {
                matrice[i][j] = valoare;
                valoare = valoare + 1;
```

```
                }
            }

        }

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                cout << matrice[i][j] <<" ";
            }
            cout << endl;
        }

        return 0;
    }
```

3.
   - a

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

using namespace std;

int main()
{
    ifstream fin("bac.in");
    ofstream fout("bac.out");
    int vectorFrecventa[10] = {0};
    int numar;
    while (fin >> numar) {
        while(numar > 0) {
            int ultimaCifra = numar % 10;
            vectorFrecventa[ultimaCifra]++;
            numar = numar / 10;
        }
    }

    for (int i = 9; i >= 0; i-- ){
        for (int j = 0; j < vectorFrecventa[i];j++) {
            fout << i;
        }
    }

    fin.close();
    fout.close();
    return 0;
}
```

   - b

Solutia este eficienta din punct de vedere al timpului de executie deoarece se efectueaza o singura parcurgere a fisierului. In acelasi timp, solutia este eficienta din punct de vedere al memoriei deoarece vom memora doar un vector de frecvente ce are doar 10 componente si nu o sa memoram toate numerele din fisier care pot fi maximum 9000000.