Subiectul I

1.
   - Aici incercam sa vedem care expresie se potriveste cerintei noastre:
     - a. nu este ok deoarece putem obtine 1 si pt numere care nu sunt divizibile cu 5 (ex x = -1 si y = 6);
     - b. Aici pare ca obtinem un rezultat adevarat doar pentru numere divizibile cu 5, deci b este raspunsul insa haideti sa vedem si celalalte
     - c. Aici, putem obtine 1 si pentru numere care nu sunt divizibile cu 5 (ex x = 1 si y = 9)
     - d. Aici putem obtine 1 pentru x = 3 si y=10, deci contrar cerintei noastre.
   - Raspuns corect: b
2.
   - Rezolvare:

```
f(2023) =
        = 1 + f(505) =
            = 1 + f(126)
                = 1 + f(31)
                    = 1 + f (7)
                        = 0
                    = 1
                = 2
            = 3
        = 4
```

   - Raspuns corect d
3.
   - Rezolvare:

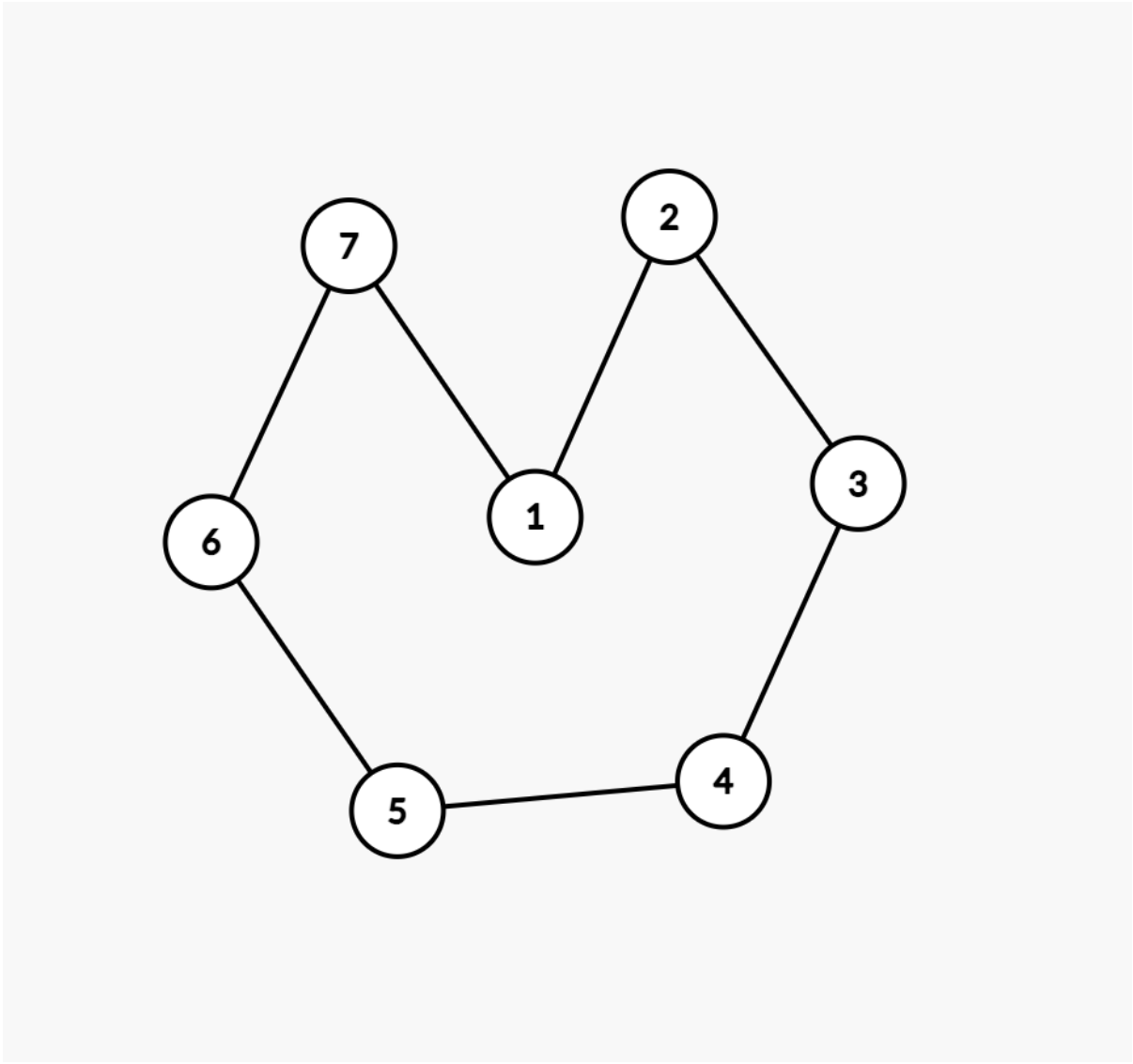
```
Numerele generate sunt
    1 2 3 4 5
    1 2 3 4 6
    1 2 3 4 7
    1 2 3 4 8
    1 2 3 4 9
    1 2 3 5 6
    1 2 3 5 7
    1 2 3 5 8
    1 2 3 5 9
    1 2 4 5 6
    1 2 4 5 7
    1 2 4 5 8
    1 2 4 5 9
    1 2 5 6 7
    1 2 5 6 8
    1 2 5 6 9
    1 2 5 7 8
    1 2 5 7 9
    1 2 5 8 9
    1 2 6 7 8
    1 2 6 7 9
```

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

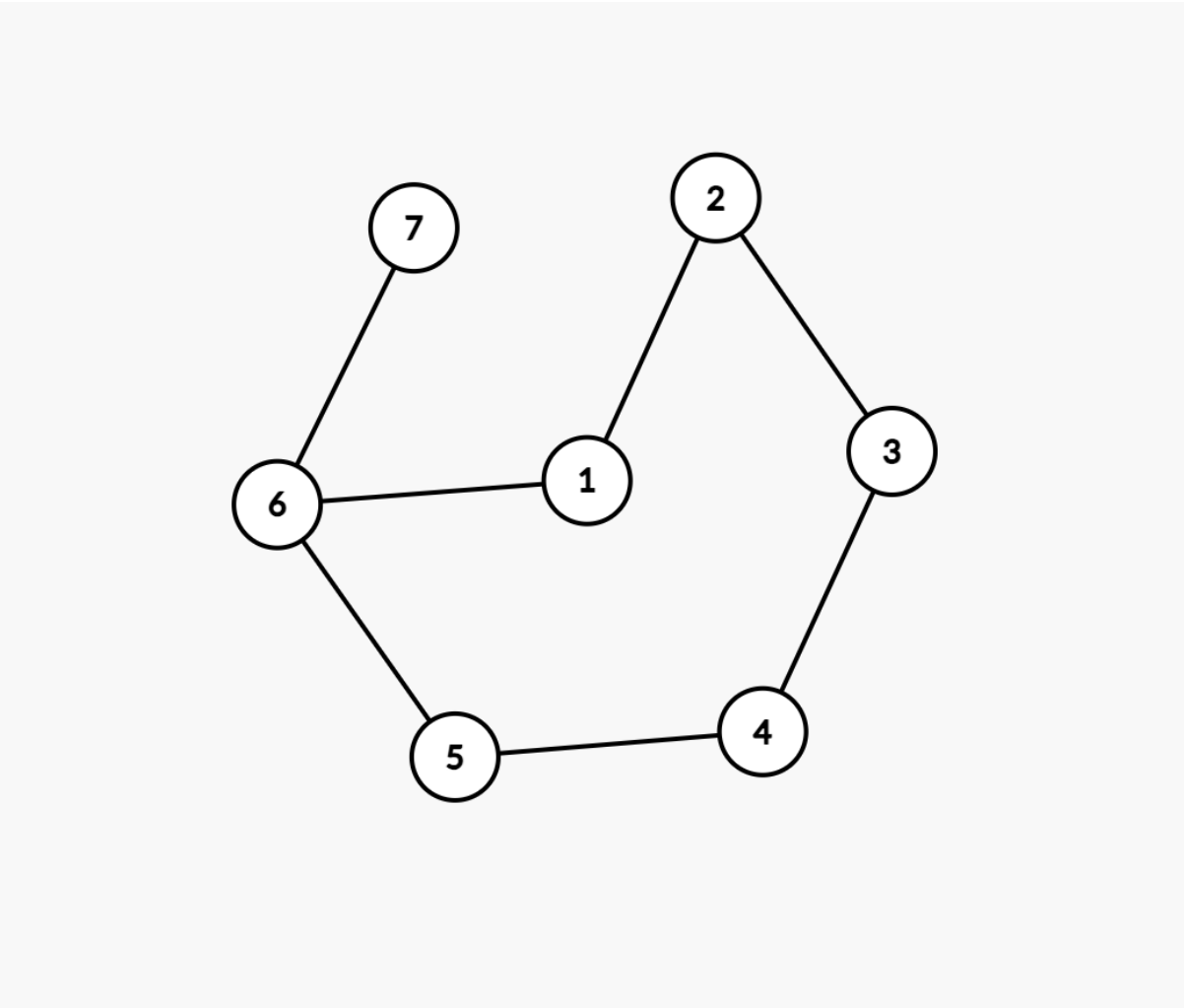
- Raspuns corect: b

4. Rezolvare:

- Poza graf initial



- Poza dupa eliminare (muchie 1,7) si adaugare muchie (1,6):



- Inainte de a explica solutia hai sa vedem fiecare ce inseamna:
  - `Graf eulerian`: Se numește graf eulerian un graf care conține un ciclu eulerian. Se numește ciclu eulerian un ciclu care conține toate muchiile grafului.
  - `Graf hamiltonian`: Se numește graf hamiltonian un graf care conține un ciclu hamiltonian. Se numește ciclu hamiltonian un ciclu elementar care conține toate vârfurile grafului.
  - `Graf aciclic`: Un graf neorientat care nu conține niciun ciclu se numește aciclic. Se numește `ciclu` un lanț simplu în care primul vârf este identic cu ultimul
  - `Graf conex`: Un graf neorientat se numește graf conex dacă pentru oricare două vârfuri x și y diferite ale sale, există cel puțin un lanț care le leagă, adică x este extremitatea inițială și y este extremitatea finală.
- Dupa cum putem observa, graf-ul dat este doar un grad conex, acum hai sa vedem de ce nu sunt bune celelalte variante:
  - Un graf G = (X,U), fără vârfuri izolate, este eulerian dacă şi numai dacă este conex şigradele tuturor vârfurilor sale sunt numere pare. Noi avem doua noduri cu grad impar (7 si 6)
  - De asemenea, graful nu este hamiltonian pentru ca nu avem un circuit care sa treaca prin toate node-urile fara sa repete unul dintre nod-uri
  - Graful nu este aciclic deoarece avem cel putin un ciclu: 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1 -> 6 -> 7
- In concluzie, raspuns corect: `d`

5. - Rezolvare:

- Dacă într-un arbore binar numărul nodurilor terminale este a, iar c este numărul nodurilor care au exact 2 fii, atunci a = c + 1.
- In concluzie, raspuns corect: `2023` adica `d`
- Nota: poti reviziui mai multe teoreme si proprietati ale arborilor binari aici: https://www.pbinfo.ro/articole/25641/arbori-binari#intlink-1

Subiectul II

1.
   - Rezolvare:
   - a.

```
x = 2378, y = 503
n = 0, p = 1
    8 > 3
        z = 8
    n = 0 + 8 * 1 = 8
    p = 10
    x = 237
    y = 50
237 != 0 && 50 != 0
    7 > 0
        z = 7
    n = 8 + 7 * 10 = 78

    p = 100
    x = 23
    y = 5
23 != 0 && 5 != 0
    3 < 5
        z = 5
    n = 78 + 5 * 100 = 578
    p = 1000
    x = 2
    y = 0
2 != 0 && 0 != 0 FALS
afiseaza n => 578
```

```
 - Algoritmul afiseaza cifra cea mai mare, dintre cele doua numere,
mergand de la dreapta la stanga, cat timp sunt cifre de parcurs in
ambele numere.  Adica, o sa comparam ultima cifra din x cu ultima
cifra din y, o alegem pe cea mai mare, mergem si comparam penultima
cifra din x cu penultima cifra din y, si alegem pe cea mai mare, etc.
```

   - b `223` si `10`
   - c

```
#include <iostream>

using namespace std;

int main()
{
    int x, y;
    cin >> x >> y;
    int n = 0, p = 1;
    do {
        int z;
        if (x % 10 > y % 10) {
            z = x % 10;
        } else {
            z = y % 10;
        }
        n = n + z * p;
        p = p * 10;
        x = x / 10;
        y = y / 10;
    } while (x != 0 && y != 0);

    cout << n;

    return 0;
}
```

- d

```
citeste x,y;
n <- 0; p <- 1;
repeta
    daca(x%10 > y%10)
        atunci z <- x%10;
    altfel
        z <- y % 10;
    n <-  n + z*p;
    p <- p * 10;
    x <-  [x/10];
    y <- [y/10]
pana cand  (x == 0 sau y == 0);

scrie n
```

2.

```
c = "examen"
i = 0
    c[0] = c[1] // c = xxamen
```

```
    i = 1
        c[1] = c[2] // c = xaamen
    i = 2
        c[2] = c[3] // c = xammen
```

3.

- Rezolvare

```
    - Stiva functioneaza pe principiul ultimul venit, primul servit.
    - Coada functioneaza pe principiul primul venit, primul servit
    - Stiva va contine:
        2023 - varf
        2022 - pozitia 2
        2021 - pozitia 3
    - Coada va contine:
        2026 - primul element
        2025 - al doilea element
        2024 - al treilea element
    - Dupa ce se extrag toate elementele din coada, si se pun in
stiva, elementul din varful stivei va fi ultimul element din  coada,
adica 2024
```

Subiectul III

1. Rezolvare:

```
    int suma(int v[], int n) {
        int rezultat = 0;
        for (int i = 0; i < n; i++) {
            if (v[i] % 2022 == 0 || v[i] % 2022 == 1 || v[i] % 2022 ==
2) {
                rezultat += v[i];
            }
        }

        return rezultat;
    }
        ```
```

2. Rezolvare
    - Nota: cautarea binara presupune ca setul de date in care cautam sa fie sortat crescator.
    - Ideea simplificata este ca mereu ne raportam la elementul ce se afla in mijlocul sirului.
        - Daca valoarea cautata este mai mare decat mijlocul sirului, o sa continuam sa impartim in doua partea din stanga a sirului unde vom compara din nou cu mijlocul acesteia, etc
```

- Daca valoarea cautata este mai mica decat mijlocul sirului, o sa continuam sa impartim in doua partea din dreapta a sirului unde vom compara din nou cu mijlocul acestia, etc
- Daca valoarea cautata este egala cu mijlocul, am gasit ce cautam.

```
    #include <iostream>

    using namespace std;

    int suma(int v[], int n);

    int main()
    {
        int n = 4, x = 20;
        int clase[4] = {16, 18, 20, 23};
        int stanga = 0, dreapta = n -1;
        int pozitie = -1; // initializam cu -1 pozitia unde gasim
numarul de carti potrivite pentru clasa ceruta.
        while (stanga <= dreapta) {
            int mijloc = (stanga + dreapta) / 2;
            if ( x <  clase[mijloc]) {
                dreapta = mijloc - 1;
            } else if (x > clase[mijloc]) {
                stanga= mijloc + 1;
            } else {
                pozitie = mijloc;
                break;
            }
        }

        if (pozitie == -1) {
            cout << "NU";
        } else {
            cout << "DA";
        }
        return 0;
    }
```

3. Rezolvare:
   - a:
     - Cautam puterile lui 3 care pana cand obtinem o putere mai mica sau egala cu 3.
     - Daca puterea gasita este mai mare sau egala cu x, atunci o afisam, altfel afisam 0.
     - Programul este eficient din puncte de vedere al timpului de executie deoarece parcurgem o singura data fisierul. In acelasi timp, programul este eficient din punct de vedere al memoriei utilizate deoarece tinem in memorie, doar capetele intervalului, cate 2 pe rand.
   - b:

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

```cpp
using namespace std;

int main()
{
    ifstream fin("bac.txt");
    int x, y;
    while (fin >> x && fin >> y) {
        int putere = 1;
        while (putere * 3 <= y) {
            putere = putere * 3;
        }
        if (putere >= x) {
            cout << putere << " ";
        } else {
            cout << 0 << " ";
        }
    }

    return 0;
}
```