Laborator 6

Problema 6

Scrieti o clasa template care gestioneaza elementele de pe o coada.

main.cpp

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

// Definim capacitatea maxima pentru coada
#define SIZE 40

// Clasa pentru coada
template <class X>
class queue
{
    X *arr;         // Array pentru a tine elementele din coada
    int capacity;   // Capacitatea maxima pentru coada
    int front;      // Front pointuieste primul element din coada
    int rear;       // Rear pointuieste la ultimul element din coada
    int count;      // Marimea curenta a cozii

public:
    queue(int size = SIZE); // Constructorul cozii
    void dequeue(); // Elimina un element
    void enqueue(X x); // Adauga un element
    X peek(); // Arata primul element din lista
    int size(); // Arata dimentiunea listei
    bool isEmpty(); // Arata daca coada e goala sau nu
    bool isFull(); // Arata daca coada e plina sau nu
};

// Constructor pentru a initializa clasa
template <class X>
queue<X>::queue(int size)
{
    arr = new X[size];
    capacity = size;
    front = 0;
    rear = -1;
    count = 0;
}

// Functie pentru a scoate un element din coada
template <class X>
void queue<X>::dequeue()
{
    // Verificam daca mai sunt elemente de sters in coada
    if (isEmpty())
```

```
    {
        cout << "Nu mai sunt elemente in coada\nProgram terminat\n";
        exit(EXIT_FAILURE);
    }

    cout << "Stergere " << arr[front] << '\n';

    front = (front + 1) % capacity;
    count--;
}

// Functie pentru a adauga un element in coada
template <class X>
void queue<X>::enqueue(X item)
{
    // Verificam daca coada este plina si daca mai putem adauga sau nu elemente
    if (isFull())
    {
        cout << "Coada este plina\nProgram terminat\n";
        exit(EXIT_FAILURE);
    }

    cout << "Se adauga " << item << '\n';

    rear = (rear + 1) % capacity;
    arr[rear] = item;
    count++;
}

// Functie pentru a returna primul element din coada
template <class X>
X queue<X>::peek()
{
    if (isEmpty())
    {
        cout << "Coada este goala\nProgram terminat\n";
        exit(EXIT_FAILURE);
    }
    return arr[front];
}

// Functie pentru a returna dimensiunea cozii
template <class X>
int queue<X>::size()
{
    return count;
}

// Functie pentru a returna daca coada este goala sau nu
template <class X>
bool queue<X>::isEmpty()
{
```

```
    return (size() == 0);
}

// Functie pentru a returna daca coada este plina sau nu
template <class X>
bool queue<X>::isFull()
{
    return (size() == capacity);
}

int main()
{
    // Creeam o coada de dimensiune 4
    queue<string> q(4);

    q.enqueue("a");
    q.enqueue("b");
    q.enqueue("c");

    cout << "Elementul din capatul cozi este: " << q.peek() << endl;
    q.dequeue();

    q.enqueue("d");

    cout << "Dimendiunea cozii este " << q.size() << endl;

    q.dequeue();
    q.dequeue();
    q.dequeue();

    if (q.isEmpty())
        cout << "Coada este goala\n";
    else
        cout << "Coada nu este goala\n";

    return 0;
}
```

output

```
Se adauga a
Se adauga b
Se adauga c
Elementul din capatul cozi este: a
Stergere a
Se adauga d
Dimendiunea cozii este 3
Stergere b
Stergere c
Stergere d
Coada este goala
Program ended with exit code: 0
```