LAB-04

Name: K V Jaya Harsha

Roll no: CS23B1034 Date: 21-08-2024

Q1. Linear Queue. (in cpp)

```cpp
// CS23B1034
// K V Jaya Harsha
#include <iostream>
using namespace std;
class Queue{
    int front, rear, capacity;
    int *queue;
public:
    Queue(int cap){
        front = -1;
        rear = -1;
        capacity = cap;
        queue = new int[cap];
    }
    ~Queue(){
        delete[] queue;
    }
    void enque(int data){
        if (rear == capacity - 1){
            cout << "Queue Overflow" << endl;
            return;
        }
        if (front == -1){
            front = 0;
        }
        queue[++rear] = data;
    }
    void dequeue(){
        if (front == -1 || front > rear){
            cout << "Queue Underflow" << endl;
            return;
        }
        for(int i = 0; i < rear; i++){
            queue[i] = queue[i + 1];
        }
        rear--;
        if (rear < front){
            rear = -1;
            front = -1;
        }
    }
    void display(){
        if (front == -1 || front > rear){
            cout << "Queue Underflow -- it is empty.";
            return;
        }
        for (int i = front; i <= rear; i++){
            cout << queue[i] << " ";
        }
        cout << endl;
    }
    void frontelement(){
        if (front == -1 || front > rear){
            cout << "Queue Underflow -- it is empty.";
            return;
        }
        cout << "Front element is " << queue[front] << ". " << endl;
    }
};
```

```
int main(){
    Queue q1(10);
    q1.dequeue();
    q1.enque(10);
    q1.enque(20);
    q1.enque(30);
    q1.enque(40);
    q1.dequeue();
    q1.dequeue();
    q1.dequeue();
    q1.enque(10);
    q1.enque(20);
    q1.enque(30);
    q1.enque(40);
    q1.enque(50);
    q1.enque(60);
    q1.enque(70);
    q1.enque(80);
    q1.enque(90);
    q1.enque(100);
    q1.display();
    q1.frontelement();
    return 0;
}
```

```
Queue Underflow
Queue Overflow
40 10 20 30 40 50 60 70 80 90
Front element is 40.
```

Q2. Circular Queue. (in cpp)

```cpp
// CS23B1034
// K V Jaya Harsha
#include <iostream>
using namespace std;
class CircularQueue
{
    int front, rear, capacity, *queue;

public:
    CircularQueue(int c)
    {
        front = rear = -1;
        capacity = c;
        queue = new int[capacity];
    }
    ~CircularQueue()
    {
        delete[] queue;
    }
    void enque(int data)
    {
        if ((front == 0 && rear == capacity - 1) || (rear == (front - 1) % (capacity - 1)))
        {
            cout << "Queue Overflow" << endl;
            return;
        }
        else if (front == -1)
        {
            front = rear = 0;
            queue[rear] = data;
        }
        else if (rear == capacity - 1 && front != 0)
        {
            rear = 0;
            queue[rear] = data;
        }
        else
        {
            rear++;
            queue[rear] = data;
        }
    }
```

```
void dequeue()
{
    if (front == -1)
    {
        cout << "Queue Overflow" << endl;
        return;
    }
    if (front == rear)
    {
        front = rear = -1;
    }
    else if (front == capacity - 1)
    {
        front = 0;
    }
    else
    {
        front++;
    }
}
void display()
{
    if (front == -1)
    {
        cout << "Queue Underflow" << endl;
        return;
    }
    if (rear >= front)
    {
        for (int i = front; i <= rear; i++)
        {
            cout << queue[i] << " ";
        }
    }
    else
    {
        for (int i = front; i < capacity; i++)
        {
            cout << queue[i] << " ";
        }
        for (int i = 0; i <= rear; i++)
        {
            cout << queue[i] << " ";
        }
    }
    cout << endl;
}
```

```
    void frontElement()
    {
        if (front == -1)
        {
            cout << "Queue underflow" << endl;
            return;
        }
        cout << "\nFront Element is: " << queue[front] << endl;
    }
};
int main()
{
    CircularQueue q1(10);
    q1.dequeue();
    q1.enque(10);
    q1.enque(20);
    q1.enque(30);
    q1.enque(40);
    q1.dequeue();
    q1.dequeue();
    q1.dequeue();
    q1.enque(10);
    q1.enque(20);
    q1.enque(30);
    q1.enque(40);
    q1.enque(50);
    q1.enque(60);
    q1.enque(70);
    q1.enque(80);
    q1.enque(90);
    q1.enque(100);
    q1.display();
    q1.frontElement();
    return 0;
}
```

```
Queue Overflow
Queue Overflow
40 10 20 30 40 50 60 70 80 90

Front Element is: 40
```