DSA Lab-06 Tasks

Name:	Syed Muhammad Raza Ali
Enrolment:	02-134231-028
Course:	DSA Lab
Faculty:	Miss Rabia

Lab 6: Implementation of Queue

Lab Exercises

Exercise 1: Efficient Package Management: Emily's Logistics Queue System

Emily is a manager at a small logistics company that specializes in delivering packages to local businesses. To streamline the package handling process, Emily decides to develop a C++ program using a QUEUE data structure with a maximum capacity of 6 packages.

The program will include three key functions:

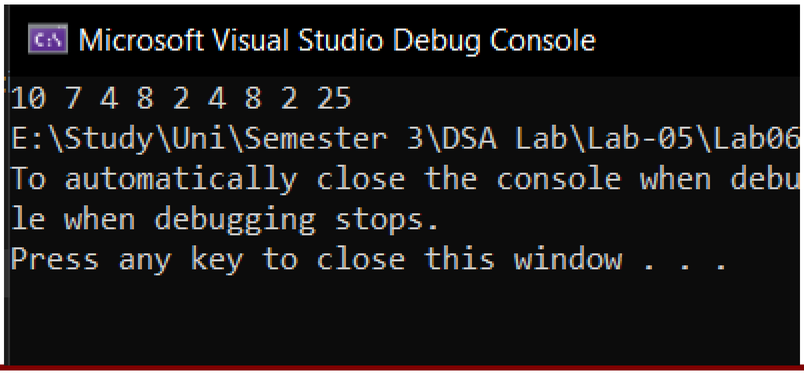
1. enQueue() to add packages to the delivery queue,
2. deQueue() to remove packages once they are delivered, and
3. Display() to show the current status of the delivery queue.

Emily plans to test the program with a series of procedures based on the company's typical operations:

1. Initially, Emily calls enQueue(10) to add a package labeled as "Package 10" to the delivery queue.
2. Following that, she adds more packages by calling enQueue(7), enQueue(4), enQueue(8), enQueue(2), and enQueue(15), each time specifying the package number.
3. Curious to see the current status of the delivery queue, Emily calls the Display() function.
4. As more packages arrive at the warehouse, Emily decides to add one more by calling enQueue(25).
5. Once the delivery drivers start their routes, Emily begins removing packages from the queue by calling deQueue() twice to signify that the first two packages have been delivered successfully.
6. To ensure that everything is running smoothly, Emily calls Display() again to check the updated status of the delivery queue after the deliveries.

Code:

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


void enQue(int queue[],int maxsize,int&front,int&rear,int item) {
	if (rear == maxsize)
		cout << "Queue is Full" << endl;
	else {
		if (front == -1)
			front = 0;
		rear = rear + 1;
		queue[rear] = item;
	}
}
void deQue( int& front, int& rear) {
	if (front == -1 or front == rear + 1)
		cout << "Queue is empty" << endl;
	else
		front += 1;
}
void displayQueue(int queue[],int&front,int&rear) {
	for (int i = front; i <= rear; i++)
		cout << queue[i] << " ";
}

int main() {

	int myQueue[6];
	int maxsize = 5;
	int front = -1;
	int rear = -1;

	enQue(myQueue, maxsize, front, rear, 10);
	enQue(myQueue, maxsize, front, rear, 7);
	enQue(myQueue, maxsize, front, rear, 4);
	enQue(myQueue, maxsize, front, rear, 8);
	enQue(myQueue, maxsize, front, rear, 2);

	displayQueue(myQueue,front,rear);

	enQue(myQueue, maxsize, front, rear, 25);
	deQue(front,rear);
	deQue(front, rear);
	displayQueue(myQueue, front, rear);

	return 0;
}
```



```
Microsoft Visual Studio Debug Console
10 7 4 8 2 4 8 2 25
E:\Study\Uni\Semester 3\DSA Lab\Lab-05\Lab06
To automatically close the console when debu
le when debugging stops.
Press any key to close this window . . .
```

Exercise 2: Ticket Booking System for a Movie Theater

Alice, the manager of a popular movie theater, is seeking to streamline the ticket booking process for her customers. To achieve this, she plans to develop a C++ program using a Circular QUEUE data structure with a maximum capacity of 6 tickets.

The program will include three key functions:

1. enQueue() to add tickets to the booking system,
2. deQueue() to remove tickets once they are booked,
3. and Display() to show the current availability of tickets.

Alice intends to test the functionality of her program with a series of procedures reflecting typical movie theater operations:

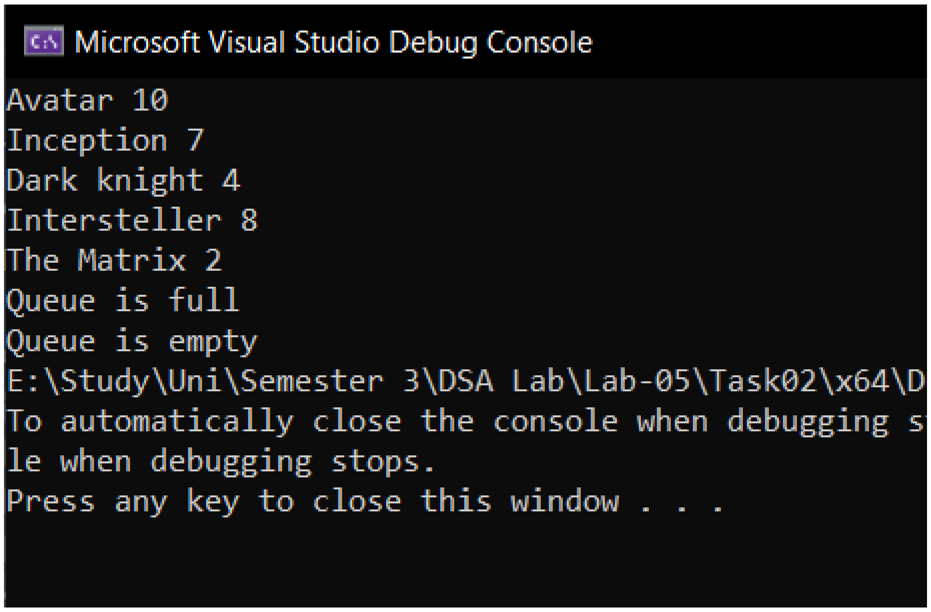
Code:

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

struct Movie {
	string nameOfMovie;
	int numberOfTickets;
}movieObj[6];

void enQueue(Movie obj[], int maxsize, int& front, int& rear, string nameOfMovie,int
numberOfTickets) {
	if (front == rear + 1 or (front == 0 and rear == maxsize))
		cout << "Queue is full" << endl;
	else {
		if (front == -1) {
			front = 0;
			rear = 0;
		}
		else if(rear == maxsize and front !=0)
			rear = 0;
```

```
            else
                  rear += 1;
            obj[rear].nameOfMovie = nameOfMovie;
            obj[rear].numberOfTickets = numberOfTickets;
      }
}
void deQueue(Movie obj[], int maxsize, int& front, int& rear) {
      if (front == -1)
            cout << "Queue is empty";
      else {
            if (front == rear) {
                  front = -1;
                  rear = -1;
            }
            else if (front == maxsize)
                  front = 0;
            else
                  front += 1;
      }
}
void displayQueue(Movie obj[],int&front,int&rear) {
      for (int i = front; i < rear; i++)
            cout << obj[i].nameOfMovie << " " << obj[i].numberOfTickets << endl;
}
int main() {


      int maxsize = 5;
      int front = -1;
      int rear = -1;

      enQueue(movieObj, maxsize, front, rear, "Avatar", 10);
      enQueue(movieObj, maxsize, front, rear, "Inception", 7);
      enQueue(movieObj, maxsize, front, rear, "Dark knight", 4);
      enQueue(movieObj, maxsize, front, rear, "Intersteller", 8);
      enQueue(movieObj, maxsize, front, rear, "The Matrix", 2);
      enQueue(movieObj, maxsize, front, rear, "Star wars", 15);

      displayQueue(movieObj,front,rear);

      enQueue(movieObj, maxsize, front, rear, "Divergent", 25);

      deQueue(movieObj,maxsize,front,rear);
      deQueue(movieObj, maxsize, front, rear);
      deQueue(movieObj, maxsize, front, rear);
      deQueue(movieObj, maxsize, front, rear);
      deQueue(movieObj, maxsize, front, rear);
      deQueue(movieObj, maxsize, front, rear);
      deQueue(movieObj, maxsize, front, rear);


      displayQueue(movieObj, front, rear);
```

```
	return 0;
}
```

Output:



Microsoft Visual Studio Debug Console

```
Avatar 10
Inception 7
Dark knight 4
Intersteller 8
The Matrix 2
Queue is full
Queue is empty
E:\Study\Uni\Semester 3\DSA Lab\Lab-05\Task02\x64\D
To automatically close the console when debugging s
le when debugging stops.
Press any key to close this window . . .
```

Exercise 3: Streamlining Café Orders: Sara's Token System Implementation

Sara's café has gained immense popularity in the neighborhood for its delicious food and cozy ambiance. Recently, she introduced a token system to streamline the ordering process. The system works like this: when a customer enters the café, they take a token indicating their order and wait for their turn. As soon as their order is ready, they approach the food counter, display their token, collect their food, and pay the bill.

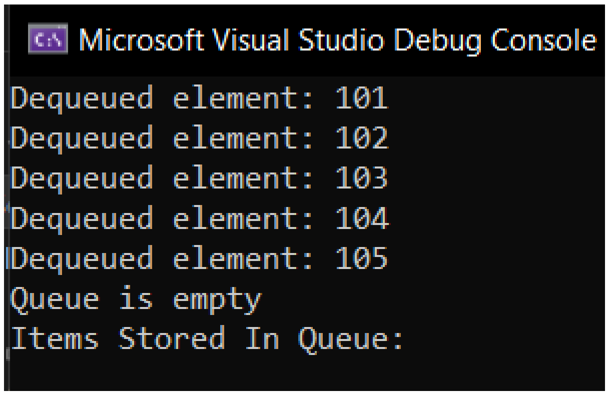
Now, Sara wants to implement a program to manage this token system efficiently. She wants to use a Circular Queue data structure to store the token numbers, with a maximum capacity of 10 tokens at any given time.

Your task is to write a C++ program that simulates this token system for Sara's café. The program should include the following functionalities:

1. A function to add tokens to the circular queue when a customer arrives at the café.
2. A function to remove tokens from the circular queue when a customer's order is ready.
3. Ensure that the circular queue maintains a maximum capacity of 10 tokens at all times.
4. Display appropriate messages to inform customers when they can collect their food.
5. Display an error message if the queue is empty and a customer tries to collect their food.

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

const int maxSize = 6;
int Queue[maxSize];
int F = -1, R = -1;

void enQueue(int item) {
    if (R == maxSize - 1) {
        cout << "Queue is full" << endl;
    }
    else {
        if (F == -1)
            F = 0;
        R++;
        Queue[R] = item;
    }
}

void deQueue() {
    if (F == -1 || F > R) {
        cout << "Queue is empty" << endl;
    }
    else {
        cout << "Dequeued element: " << Queue[F] << endl;
        F++;
    }
}

void display() {
    cout << "Items Stored In Queue: ";
    for (int i = F; i <= R; i++) {
        cout << Queue[i] << " ";
    }
    cout << endl;
}

int main() {
    enQueue(101);
    enQueue(102);
    enQueue(103);

    deQueue();
    enQueue(104);

    deQueue();
    enQueue(105);

    deQueue();
    deQueue();
    deQueue();
    deQueue();

    display();

    return 0;
}
```

Output:



Microsoft Visual Studio Debug Console

```
Dequeued element: 101
Dequeued element: 102
Dequeued element: 103
Dequeued element: 104
Dequeued element: 105
Queue is empty
Items Stored In Queue:
```