Lab 5: Understanding the Circular Queue

A. Explain the algorithm of Circular Queue.

Circular queue is a linear data structure in which the operations are performed based on FIFO principle and the last position is connected back to the first position to make a circle. It is also called 'Ring Buffer'.
Similar to normal queue, there are mainly two operation of circular queue.

1. Enqueue: Inserting a new element into the circular queue through rear position.

Algorithm:

Step 1: Start
Step 2: If (front == 0 && rear = max-1) ||
(rear ==(front-1)%(max-1))
Then write "Queue overflow" and stop.
Step 3: Read data to insert in circular queue
Step 4: Else If (front = -1)
Then set front = rear = 0 and insert Queue[rear] = data
Step 5: Else If (rear = max-1 and front != 0)
Then rear = 0 and insert Queue[rear] = data
Step 6: else
Else rear = rear + 1
Queue[rear] = data
Step 7: Stop

2. Dequeue: Deletes an element from the circular queue through front position.

Algorithm:

Step 1: Start
Step 2: If (front == -1)
Then write "Queue underflow"
Step 3: Else if(front ==rear)
Then front = rear = -1
Step 4: Else if(front == max-1)
Then front = 0

Step 5: Else Front = Front+1
Step 6: Stop

B. Why circular Queue is necessary?

In a normal queue, we can insert elements until queue becomes full. But once queue becomes full, we can not insert the next element even if there is space in front position of queue. Hence to eliminate such problem, circular queue is introduced. Thus, The unused memory locations in the case of ordinary queues can be utilized in circular queues which is very useful for memory management. CPU scheduling uses the concept of circular queue i.e operating systems often maintain a queue of processes that are ready to execute or that are waiting for a particular event to occur.

Implementation of operation of Circular Queue using C++

```
#include <iostream>
#define MAX 5
using namespace std;

class CQueue{
	int cqueue[MAX];
	int Front, Rear;
public:
	CQueue() {Front = Rear = -1;}

	int Enqueue(int item){
		if((Front == 0 && Rear == MAX-1) || (Rear == (Front-1)%(MAX-1))) cout
<< "QUEUE OVERFLOW" << endl;
		else if(Front == -1){
			Front = Rear = 0;
			cqueue[Rear] = item;
		}	//insert first element
		else if(Rear == MAX-1 && Front != 0){
			Rear = 0;
			cqueue[Rear] = item;
		}	//insert into empty space after any dequeue
		else{
			Rear++;
			cqueue[Rear] = item;
		}	//enqueue
		return 0;
```

```
    }

    int Dequeue(){
        cout << "Element deleted from queue is: " << cqueue[Front] << endl;
        if(Front == -1) cout << "QUEUE UNDERFLOW" << endl;
        if(Front == Rear) Front = Rear = -1;        //if only one element is there
        else if(Front == MAX-1) Front = 0;           //if front point to max size
                                                     //then dequeuing means
front should point to 0
                                                     //which almost similar to
empty queue
        else Front++;      //normal dequeue if not overflow condition meet
        return 0;
    }

    int Traverse(){
        if(Front == -1) cout << "QUEUE IS EMPTY" << endl;
        cout << "QUEUE ITEMS" << endl;
        if(Rear >= Front){
            for(int i = Front; i <= Rear; i++) cout << cqueue[i] << "      ";
        }
        else{
            //two conditions if Rear < Front:
            //1. Front to Max
            //2. 0 to Rear this completes the full circle
            for(int i = Front; i < MAX; i++) cout << cqueue[i] << "       ";
            for(int i = 0; i <= Rear; i++) cout << cqueue[i] << "      ";
        }
        cout << endl;
        return 0;
    }
};
int main(){
    CQueue obj1;
    int ch, val;
    cout << "1) ENQUEUE" << endl;
    cout << "2) DEQUEUE" << endl;
    cout << "3) TRAVERSE" << endl;
    cout << "4) EXIT" << endl;
    do{
        cout << "Enter choice:";
        cin >> ch;
        if(ch == 1){
            cout<<"Enter value to enqueue:" << endl;
            cin>>val;
            obj1.Enqueue(val);
        }
        else if(ch == 2)obj1.Dequeue();
        else if(ch == 3)obj1.Traverse();
    }while(ch != 4);
```

```
    return 0;
}
```