Lab 3: Understanding the Queue

A. Explain the algorithm for the operation of queue.

Queue: It is a linear data structure that serves as a container of objects that are inserted and removed according to the FIFO(First-In, First-Out) principle.

Operation of Queue:

1. **Enqueue**: Inserts a new element at the rear of the queue.

The following steps should be taken to enqueue data into a queue:

Step 1: Start
Step 2: Check if the queue is full
Step 3: If the queue is full, state overflow error and exit
Step 4: If the queue is not full, increment rear pointer to point the next empty space.
Step 5: Add data element to the queue location, where the rear is pointing.
Step 6: Stop

Algorithm:

Step 1: Initialize the front = rear=-1;
Step 2: Repeat step 3 to until rear<MAXSIZE-1
Step3: Read item
Step 4: if front==-1 then
Front=rear=0
else
rear=rear+1
step5: queue[rear]=item
step6: if condition of step 2 does not satisfy then print queue overflow.

2. **Dequeue**: Removes the front element of the queue.

The following steps should be taken to dequeue data into a queue:

Step 1: Start
Step 2: Check if the queue is empty.
Step 3: If the queue is empty, produce underflow error and exit
Step 4: If the queue is not empty, access the data where front is pointing
Step 5: Increment front pointer to point to the next available data element
Step 6: Stop

Algorithm:

Step 1: Repeat step 2 to 4 until front >= 0
Step 2: Set item=queue[front]
Step 3: If front==real
Set front=-1
Set rear=-1
else
front=front+1
Step 5: print deleted item
Step 6: print queue is empty

B. What are the difference between stack and queue?

Stack

1. It follow Last In First Out (LIFO) order i.e last inserted object is first to come out.
2. Only one pointer is used. It points to top of the stack.
3. Stacks are visualized as vertical collections.
4. Objects are inserted and removed at the same end.
5. Stack operations are called push and pop.
6. 6. Example: plates placed one above other

Queue

1. It follow First in First Out (FIFO) order i.e. object inserted first is first deleted.

2. Two pointer are used as front and rear for two ends.
3. Queue are visualized as vertical collections.
4. Objects are inserted and removed from different ends.
5. Queue operations are called enqueue and dequeue.
6. Example: queue for bus

Implementation of operation of Queue using C++

```
#include <iostream>
#define MAX 10

using namespace std;

class Queue
{
	int Front,Rear;
	int queuee[MAX];
public:
	Queue() {Front = Rear = -1;}	//constructor

	int Enqueue(int item){
		if(Rear == MAX - 1) cout << "QUEUE OVERFLOW" << endl;
		else if(Front == -1 && Rear == -1){
			Front = Rear = 0;
			queuee[Rear] = item;
			cout << "ITEM INSERTED: " << item << endl;
		}
		else{
			Rear++;
			queuee[Rear] = item;
			cout << "ITEM INSERTED: " << item << endl;
		}
		return 0;
	}
	int Dequeue(){
		int item;
		if(Rear == -1) cout << "QUEUE UNDERFLOW" << endl;
		else if(Front == 0 && Rear == 0){
			item = queuee[Front];
			Front = Rear = -1;
			cout << "ITEM DELETED: " << item << endl;
		}
		else{
			item = queuee[Front];
			Front++;
			cout << "ITEM DELETED: " << item << endl;
```

```
        }
        return 0;
    }
    int Traverse(){
        if(Front == -1) cout << "QUEUE IS EMPTY" << endl;
        else{
            cout << "QUEUE ITEMS" << endl;
            for(int i = Front; i <= Rear; i++) cout << queuee[i] << "      ";
            cout << endl;
        }
        return 0;
    }
};

int main(){
    Queue 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;

}
```