# Data Structures and Algorithms Lab

**Instructions**

Work on this lab individually. *Write **main** function first and keep on testing the functionality of each function once created.*

Program the following tasks in your **C++** compiler and then compile and execute them.

Email your solution (**.cpp**) file only to the following respective recipient till **Friday, April 09, 2021**.

**DO NOT** compress/zip your solution.

The email must be sent from your **official PUCIT email id**, otherwise it will **NOT BE ACCEPTED** and will be marked **ZERO**.

The subject of the email should be the exact name of the lab i.e. **Lab 06**. **2 MARKS** will be **DEDUCTED**, otherwise.

| Degree | Recipient Email | Subject of Email |
|---|---|---|
| BSIT Morning | dsaubt01@gmail.com | Lab 06 |
| BSIT Afternoon | dsaubt02@gmail.com | |

You are strictly not allowed to add any other data-member/constructor/function in the class. You are also not allowed to change the name or prototype of any data-member/constructor/function.

## ADT: Queue

Provide the implementation of the following generic **Queue** class; it should provide the standard circular queue structure of FIFO (First in first out) as discussed in the class.

```
template <class T>
class Queue
{
public:
	//constructor to create MAX_SIZE queue dynamically
	Queue(int MAX_SIZE);

	//destructor to free any memory resources occupied by queue
	~Queue();

	//queue manipulation operations
	void enqueue(T x);		//add an element to the rear of queue
	T dequeue();			//delete the element at the front of queue
	void clear ();			//clear the queue

	//queue status operations
	bool isEmpty(void);		//is queue empty?
	bool isFull(void);		//is queue full?

	//outputs the data in queue. If the list is empty, outputs “Empty Queue”.
	void showStructure(void);

private:
	//data members
	T *data;			//array of data items allocated dynamically based on MAX_SIZE
	int front;			//front index
	int rear;			//rear index
	const int MAX_SIZE;		//size of array queue
};
```

The show structure function must display the queue status with its front and rear pointing to the correct locations on the console.

**Sample Run:**

```
queue.Enqueue(5.0);
queue.Enqueue(6.5);
queue.showStructure();
```

```
front -->	5
		6.5	<-- rear
```

```
queue.Enqueue(-3.0);
queue.Enqueue(-8.0);
queue.showStructure();
```

```
front -->	5
		6.5
		-3
		-8	<-- rear
```

```
queue.Dequeue(a);
queue.Dequeue(a);
queue.showStructure();
```

```
front -->	-3
		-8	<-- rear
```

☺ ☺ ☺ **BEST OF LUCK** ☺ ☺ ☺