CSE225L – Data Structures and Algorithms Lab
Lab 07
Queue (array based)

In today's lab we will design and implement the Queue ADT using array.

quetype.h

```
#ifndef QUETYPE_H_INCLUDED
#define QUETYPE_H_INCLUDED

class FullQueue
{};
class EmptyQueue
{};
template<class ItemType>
class QueType
{
    public:
        QueType();
        QueType(int max);
        ~QueType();
        void MakeEmpty();
        bool IsEmpty();
        bool IsFull();
        void Enqueue(ItemType);
        void Dequeue(ItemType&);
    private:
        int front;
        int rear;
        ItemType* items;
        int maxQue;
};

#endif // QUETYPE_H_INCLUDED
```

quetype.cpp

```
#include "quetype.h"

template<class ItemType>
QueType<ItemType>::QueType(int max)
{
    maxQue = max + 1;
    front = maxQue - 1;
    rear = maxQue - 1;
    items = new ItemType[maxQue];
}
template<class ItemType>
QueType<ItemType>::QueType()
{
    maxQue = 501;
    front = maxQue - 1;
    rear = maxQue - 1;
    items = new ItemType[maxQue];
}
template<class ItemType>
QueType<ItemType>::~QueType()
{
    delete [] items;
}
template<class ItemType>
void QueType<ItemType>::MakeEmpty()
{
    front = maxQue - 1;
    rear = maxQue - 1;
}
template<class ItemType>
bool QueType<ItemType>::IsEmpty()
{
    return (rear == front);
}
template<class ItemType>
bool QueType<ItemType>::IsFull()
{
    return ((rear+1)%maxQue == front);
}
template<class ItemType>
void QueType<ItemType>::Enqueue(ItemType newItem)
{
    if (IsFull())
        throw FullQueue();
    else
    {
        rear = (rear +1) % maxQue;
        items[rear] = newItem;
    }
}
template<class ItemType>
void QueType<ItemType>::Dequeue(ItemType& item)
{
    if (IsEmpty())
        throw EmptyQueue();
    else
    {
        front = (front + 1) % maxQue;
        item = items[front];
    }
}
```

Generate the **Driver file (main.cpp)** and perform the following tasks:

Operation to Be Tested and Description of Action	Input Values	Expected Output
• Create a queue of size 5		
• Print if the queue is empty or not		Queue is Empty
• Enqueue four items	5 7 4 2	
• Print if the queue is empty or not		Queue is not Empty
• Print if the queue is full or not		Queue is not full
• Enqueue another item	6	
• Print the values in the queue		5 7 4 2 6
• Print if the queue is full or not		Queue is Full
• Enqueue another item	8	Queue Overflow
• Dequeue two items		
• Print the values in the queue		4 2 6
• Dequeue three items		
• Print if the queue is empty or not		Queue is Empty
• Dequeue an item		Queue Underflow

- Add a function **`ReplaceItem`** to the QueType class which replaces all occurrences of **`oldItem`** with **`newItem`** in the Queue.

```
void ReplaceItem(int oldItem, int newItem);
```

Sample Input &Output:

Queue Items:
21 26 13 26 29

`ReplaceItem(26, 9)` →

Queue Items:
21 9 13 9 29