CSE225L – Data Structures and Algorithms Lab
Lab 09
Queue (Linked List)

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

quetype.h

```
#ifndef QUETYPE_H_INCLUDED
#define QUETYPE_H_INCLUDED
class FullQueue
{};
class EmptyQueue
{};
template <class ItemType>
class QueType
{
    struct NodeType
    {
        ItemType info;
        NodeType* next;
    };
    public:
        QueType();
        ~QueType();
        void MakeEmpty();
        void Enqueue(ItemType);
        void Dequeue(ItemType&);
        bool IsEmpty();
        bool IsFull();
    private:
        NodeType *front, *rear;
};

#endif // QUETYPE_H_INCLUDED
```

quetype.cpp

```
#include "quetype.h"
#include <iostream>
using namespace std;

template <class ItemType>
QueType<ItemType>::QueType()
{
    front = NULL;
    rear = NULL;
}
template <class ItemType>
bool QueType<ItemType>::IsEmpty()
{
    return (front == NULL);
}
template<class ItemType>
bool QueType<ItemType>::IsFull()
{
    NodeType* location;
    try
    {
        location = new NodeType;
        delete location;
        return false;
    }
    catch(bad_alloc& exception)
    {
        return true;
    }
}
template <class ItemType>
void QueType<ItemType>::Enqueue(ItemType newItem)
{
    if (IsFull())
        throw FullQueue();
    else
    {
        NodeType* newNode;
        newNode = new NodeType;
        newNode->info = newItem;
        newNode->next = NULL;
        if (rear == NULL)
            front = newNode;
        else
            rear->next = newNode;
        rear = newNode;
    }
}
template <class ItemType>
void QueType<ItemType>::Dequeue(ItemType& item)
{
    if (IsEmpty())
        throw EmptyQueue();
    else
    {
        NodeType* tempPtr;
        tempPtr = front;
        item = front->info;
        front = front->next;
        if (front == NULL)
            rear = NULL;
        delete tempPtr;
    }
}
template <class ItemType>
void QueType<ItemType>::MakeEmpty()
{
    NodeType* tempPtr;
    while (front != NULL)
    {
        tempPtr = front;
        front = front->next;
        delete tempPtr;
    }
    rear = NULL;
}
template <class ItemType>
QueType<ItemType>::~QueType()
{
    MakeEmpty();
}
```

Generate the **Driver file (main.cpp)** and check your program with the following outputs:

Operation to Be Tested and Description of Action	**Input Values**	**Expected Output**
• 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 not Full
• Enqueue another item	8	
• Dequeue two items		
• Dequeue		
• Print the values in the queue		2 6 8
• Dequeue three items		
• Print if the queue is empty or not		Queue is Empty
• Dequeue an item		Queue Underflow
• Add a function **`Length`** to the `QueType` class which returns the number of items in the Queue. **`int Length();`** **Sample Input &Output:** Queue Items: n o w y h $\xrightarrow{\texttt{Length()}}$ Length is : 5		