North South University
CSE-225L(Data Structures & Algorithm)
Summer - 2018
Lab-11 (Queue – Linked List Based)

Class "QueType":

quetype.h

```
#ifndef QUETYPE_H_INCLUDED
#define QUETYPE_H_INCLUDED
#include <iostream>
using namespace std;


class FullQueue{};

class EmptyQueue{};

template <class DataType>
class QueType
{

	struct NodeType
	{
		DataType info;
		NodeType* next;
	};


public:
	QueType();
	~QueType();
	void MakeEmpty();
	void Enqueue(DataType);
	DataType Dequeue();
	bool IsEmpty();
	bool IsFull();
private:
	NodeType *front, *rear;
};
#endif // QUETYPE_H_INCLUDED
```

quetype.cpp

```
#include "quetype.h"

template <class DataType>
QueType<DataType>::QueType()
{
	front = NULL;
	rear = NULL;
}
```

```
template <class DataType>
bool QueType<DataType>::IsEmpty()
{
          return (front == NULL);
}


template<class DataType>
bool QueType<DataType>::IsFull()
{
     NodeType* location;
     try
     {
          location = new NodeType;
          delete location;
          return false;
     }
     catch(bad_alloc& exception)
     {
          return true;
     }
}

template <class DataType>
void QueType<DataType>::Enqueue(DataType 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 DataType>
DataType QueType<DataType>::Dequeue()
{

          DataType item;

          if (IsEmpty())
               throw EmptyQueue();
          else
          {

               NodeType* tempPtr;
               tempPtr = front;

               item = front->info;

               front = front->next;

               if (front == NULL)
                    rear = NULL;
               delete tempPtr;

               return item;
          }
}

template <class DataType>
void QueType<DataType>::MakeEmpty()
{
     NodeType* tempPtr;

     while (front != NULL)
     {
          tempPtr = front;
          front = front->next;
          delete tempPtr;
     }
     rear = NULL;
}

template <class DataType>
QueType<DataType>::~QueType()
{
          MakeEmpty();
}
```