North South University
CSE-225L(Data Structures & Algorithm)
Summer - 2018
Lab-13 (Priority Queue)

Class "HeapType":

heaptype.h

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

template<class ItemType>
struct HeapType
{
    void ReheapDown(int root,int bottom);
    void ReheapUp(int root,int bottom);
    ItemType* elements;
    int numElements;
};
#endif // HEAPTYPE_H_INCLUDED
```

heaptype.cpp

```
#include "heaptype.h"

template<class ItemType>
void Swap(ItemType& one, ItemType& two)
{
    ItemType temp;
    temp = one;
    one = two;
    two = temp;
}

template<class ItemType>
void HeapType<ItemType>::ReheapDown(int root, int bottom)
{
    int maxChild;
    int rightChild;
    int leftChild;

    leftChild = root*2+1;
    rightChild = root*2+2;

    if(leftChild <= bottom)
    {
        if(leftChild == bottom)
            maxChild = leftChild;
```

```
        else
        {
            if(elements[leftChild]<=elements[rightChild])
                maxChild = rightChild;
            else
                maxChild = leftChild;
        }

        if(elements[root] < elements[maxChild])
        {
            Swap(elements[root], elements[maxChild]);
            ReheapDown(maxChild, bottom);
        }
    }
}


template<class ItemType>
void HeapType<ItemType>::ReheapUp(int root, int bottom)
{
    int parent;
    if(bottom>root)
    {
        parent = (bottom-1)/2;
        if (elements[parent] < elements[bottom])
        {
            Swap(elements[parent], elements[bottom]);
            ReheapUp(root, parent);
        }
    }
}

template class HeapType<int>;
template class HeapType<float>;
template class HeapType<double>;
template class HeapType<long>;
template class HeapType<short>;
template class HeapType<char>;
```

Class "PQType":

pqtype.h

```
#ifndef PQTYPE_H_INCLUDED
#define PQTYPE_H_INCLUDED
```

```
#include "heaptype.h"

class FullPQ{};

class EmptyPQ{};

template<class ItemType>
class PQType
{
    public:

        PQType(int);
        ~PQType();
        void MakeEmpty();
        bool IsEmpty();
        bool IsFull();
        void Enqueue(ItemType);
        void Dequeue(ItemType&);

    private:

        int length;
        HeapType<ItemType> items;
        int maxItems;
};
#endif // PQTYPE_H_INCLUDED
```

pqtype.cpp

```
#include "pqtype.h"

template<class ItemType>
PQType<ItemType>::PQType(int max)
{
    maxItems = max;
    items.elements=new ItemType[max];
    length = 0;
}

template<class ItemType>
PQType<ItemType>::~PQType()
{
    delete [] items.elements;
}

template<class ItemType>
void PQType<ItemType>::MakeEmpty()
```

```
{
    length = 0;
}

template<class ItemType>
bool PQType<ItemType>::IsEmpty()
{
    return length == 0;
}

template<class ItemType>
bool PQType<ItemType>::IsFull()
{
    return length == maxItems;
}

template<class ItemType>
void PQType<ItemType>::Enqueue(ItemType newItem)
{
    if (length == maxItems)
        throw FullPQ();
    else
    {
        length++;
        items.elements[length-1] = newItem;
        items.ReheapUp(0, length-1);
    }
}

template<class ItemType>
void PQType<ItemType>::Dequeue(ItemType& item)
{
    if (length == 0)
        throw EmptyPQ();
    else
    {
        item = items.elements[0];
        items.elements[0] =
        items.elements[length-1];
        length--;
        items.ReheapDown(0, length-1);
    }
}

template class PQType<int>;
template class PQType<float>;
template class PQType<double>;
template class PQType<long>;
```

```
template class PQType<short>;
template class PQType<char>;
```