CSE225L – Data Structures and Algorithms Lab
Lab 14
Priority Queue

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

heaptype.h

```
#ifndef HEAPTYPE_H_INCLUDED
#define HEAPTYPE_H_INCLUDED
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);
        }
    }
}
```

pqtype.h

```
#ifndef PQTYPE_H_INCLUDED
#define PQTYPE_H_INCLUDED
#include "heaptype.h"
#include "heaptype.cpp"
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);
    }
}
```

Now generate the **Driver file (main.cpp)** where you perform the following tasks:

Operation to Be Tested and Description of Action	**Input Values**	**Expected Output**
• Add a member function `PrintQueue` to the `PQType` class which prints the content of the heap		
• Create a `PQType` object		
• Print if the queue is empty or not		Queue is empty
• Insert ten items, in the order they appear	4 9 2 7 3 11 17 0 5 1	
• Print if the queue is empty or not		Queue is not empty
• Print the elements in the heap		17 7 11 5 3 2 9 0 4 1
• Dequeue one element and print the dequeued value		17
• Dequeue one element and print the dequeued value		11
• Print the elements in the heap		9 7 4 5 3 2 1 0
• Dequeue three more elements		
• Print the elements in the heap		4 3 2 0 1
• Modify the `ReheapUp` and the `ReheapDown` functions in such a way that the `PQType` class now works as a min-heap		
• Insert ten items, in the order they appear	4 9 2 7 3 11 17 0 5 1	
• Print the elements in the heap		0 1 4 3 2 11 17 9 5 7
• Dequeue one element and print the dequeued value		0
• Dequeue one element and print the dequeued value		1
• Print the elements in the heap		2 3 4 5 7 11 17 9
• Dequeue three more elements		
• Print the elements in the heap		5 7 11 9 17