CISC 2200 - Data Structures (Fall, 2022) Prof. Rahouti

Homework 5

100 points

I. Priority Queues & Heaps (100 pts)

As you have seen in class, priority queues can be implemented using heaps along with necessary operations to maintain and restore the heap properties.

Consider the following skeleton code for the *heap.h* and *PQType.h* files. You are required to implement the declared functions in both *heap.cpp* and *PQType.cpp* files separately. Precisely:

a. Swap \\ heap.h file (10 pts)
b. ReheapUp \\ heap.h file (10 pts)
c. RehaepDown \\ heap.h file (10 pts)
d. PQType(int); \\ PQType.h (10 pts)
e. ~PQType(); \\ PQType.h (10 pts)
f. MakeEmpty(); \\ PQType.h (10 pts)
g. IsEmpty(); \\ PQType.h (10 pts)
h. IsFull(); \\ PQType.h (10 pts)
i. Enqueue(ItemType newItem); \\ PQType.h (10 pts)
j. Dequeue(ItemType& item); \\ PQType.h (10 pts)

A. *heap.h/cpp* files

```
//-----------------------------------------------------------------------
template <class ItemType>
void swap(ItemType& one, ItemType& two);


template<class ItemType>
// Assumes ItemType is either a built-in simple type or a class
// with overloaded relational operators.
struct HeapType{
  void ReheapDown(int root, int bottom);
  void ReheapUp(int root, int bottom);
  ItemType* elements;   // Array to be allocated dynamically
  int numElements;
};

template <class ItemType>
void Swap(ItemType& one, ItemType& two)
{
//………………….
}

template<class ItemType>
void HeapType<ItemType>::ReheapUp(int root, int bottom)
{
//…………………………..
}

template<class ItemType>
void HeapType<ItemType>::ReheapDown(int root, int bottom)
{
//………………………………
}
//--------------------------------------------------------------------------------
```

B. *PQType.h/cpp* file

```
class FullPQ{};
class EmptyPQ{};
#include "heap.h"
template<class ItemType>
class PQType
```

```
{
public:
  PQType(int);
  ~PQType();
  void MakeEmpty();
  bool IsEmpty() const;
  bool IsFull() const;
  void Enqueue(ItemType newItem);
  /* Function: Adds newItem to the rear of the queue. if (the priority queue is full) exception
FullPQ is thrown; else newItem is in the queue */
  void Dequeue(ItemType& item);
  /* Function: Removes element with highest priority from the queue and returns it in item. If (the
priority queue is empty) exception EmptyPQ is thrown; else highest priority element has been
removed from queue. item is a copy of removed element */

private:
  int length;
  HeapType<ItemType> items;
  int maxItems;
};
```