COP 3530 Fall 2013

Data Structures and Algorithms

Assignment 5. Due: Monday, Nov 4, 11:55 p.m. via Sakai

In this problem you will create a Min Priority Queue that is implemented as a min heap, and perform binary tree traversals and sorting of a simple list of numbers.

The main points we shall be covering are:

* Continue using ArrayList based data structures, last time …
* Heaps, Binary Trees, Priority Queues
* Usages of makefiles / make utility in UNIX/Linux
* Use of abstract data types in C++, separate compilation
* Use of header files and libraries for ArrayList

Priority queue, min heaps, and max heaps have discussed in class. A heap is a complete binary tree. A min heap (a heap in which every node has a value that is <= its kids) is our choice of implementation the Min Priority Queue.

You will implement a Min Priority Queue based on Min Heap. Min heap will in turn be implemented based on Binary Tree, and Binary Tree will be implemented based on Arraylist. You will first insert the elements into the Min Priority Queue, and then print the numbers as they are removed from the Priority Queue, as in Lecture 23, slides 6-12. NOTE: the slides show the elements being removed from the Priority Queue and stored into an array. You should not store them anywhere as they are removed from the Priority Queue. Instead, merely print them.

You will accomplish the following tasks:

1. Insert the ten elements into the Min Priority Queue (see main.cpp). The algorithm to perform the insertions (see initialize in main.cpp) must be O(n), i.e. do not perform ten insert operations on the Priority Queue, as that would cost O(n log n). Instead, initialize the Min Priority Queue by initializing the underlying min heap, using the algorithm in Lecture 24.
2. Print out the elements in the tree using:
   * Preorder
   * Postorder
   * Inorder
3. Lastly you remove (and print) all items from the Priority Queue. As they are removed, they should come out in ascending order.

You need to write (at a minimum) the following methods:

1. initialize
2. inOrder
3. preOrder
4. postOrder
5. sort

Files provided by instructor: (see the Announcements page)

* main.cpp. Creates the structure and calls the methods.
* Linearlist.h, Arraylist.cpp, and Arraylist.h. Same as in Assignment 4.
* myException.cpp and myException.h. Same as in Assignment 4.

Files you must develop

* BinaryTree.h and BinaryTree.cpp  (uses ArrayList)
* MinHeap.h and MinHeap.cpp (uses BinaryTree)
* PriorityQueue.h and Priority.cpp to sort (uses MinHeap)
* Makefile. Each .cpp file must compiled separately. No templates. Be sure to provide “clean” and ‘tar” targets in the make file.

The expected output is as follows:

thunder:5% make

g++ -c main.cpp

g++ -c MinHeap.cpp

g++ -c myException.cpp

g++ -c Arraylist.cpp Arraylist.h

g++ -c BinaryTree.cpp

g++ -c PriorityQueue.cpp

g++ main.o MinHeap.o myException.o Arraylist.o BinaryTree.o PriorityQueue.o -o main

thunder:6% ./main

The inorder travesal is: 10 3 7 2 9 6 1 5 4 8

The preorder travesal is: 1 2 3 10 7 6 9 4 5 8

The postorder travesal is: 10 7 3 9 6 2 5 8 4 1

The sorted list is: 1 2 3 4 5 6 7 8 9 10

• Your submission MUST run on thunder.cise.ufl.edu. That is where it will be run and graded.

The simplest syntax of the tar command is as follows.

tar cvf (tar\_file\_name) (file 1) (file 2) (file 3)...

To extract the contents of a tar file:

tar xvf (tar\_file\_name)

As before, use the Makefile to create the tarfile.

It is STRONGLY recommended you verify your submission is successful by downloading it from Sakai (into a separate directory), untarring it, and running make.