CS 260
Programming Assignment 7

This lab is worth a total of 100 points; 10 points are the self-evaluation, 50 points for the basic lab, 30 points for the advanced lab, and 10 points for a solution to the thinking problem.

Base Lab Specification

Build a class Heap and a class PriorityQueue. Reference the documents in Moodle for your lab. PriorityQueue should use class composition and include an instance of Heap.

This lab should follow the course coding requirements and be split into multiple files as per your chosen language. There is a driver provided for this lab, you should test each part of your code as it is developed.

There are two deliverables for this part of the lab:

class Heap

Creates a Heap using an array. The values are stored at locations 1 – N as described in the document on Heaps. You will need to code trickle-up and bubble-down algorithms.

- constructor (init) – creates an array equal to the parameter passed in (default 10)
- addItem – adds a new item to the heap
- getItem – removes and returns the smallest item from the heap
- heap should throw exception on take from empty and double when full
- should be defined in Heap.cpp/Heap.h(hpp) or Heap.py or Heap.cs

class PriorityQueue

Create a PriorityQueue using a Heap (class composition). Use the Heap methods to implement your PriorityQueue.

- constructor (init) – creates and uses heap of the proper size (default 10)
- addItem – adds a new item to the priority queue
- getItem – removes and returns the smallest from the priority queue
- should be defined in PriorityQ.cpp/PriorityQ.h(hpp) or PriorityQ.py or PriorityQ.cs

Advanced Lab Specification

For the advanced lab, create the three sorts described below. They are not classes, just functions. Each is passed an array that is sorted and returned via the reference parameter. They may need helper functions, depending on how they are implemented. For C#, they should be defined as a static class. For C++ and Python, they are functions defined in a separate file or module.

Heapsort

Code a heapsort that sorts an array from smallest at location 0 to largest. Does not use Heap Class.

- function heapSort(array, length)
- sorts the array using heapsort from smallest to largest
- Should be defined in recSorts.cpp/recSorts.h or recSorts.py or recSorts.cs

Mergesort

Code a mergesort that sorts an array from smallest at location 0 to largest.

- function mergeSort(array, length)
- sorts the array using mergesort from smallest to largest
- should be defined in recSorts.cpp/recSorts.h or recSorts.py or recSorts.cs

Quicksort

Code a quicksort that sorts an array from smallest at location 0 to largest

- function quicksort(array, length)
- sorts the array using quicksort from smallest to largest
- should be defined in recSorts.cpp/recSorts.h or recSorts.py or recSorts.cs

Thinking Problem

For the thinking problem, create a modified version of Quicksort that returns the value at the Nth index of the array, without doing a complete sort.

- function findNth(array, length, n)
- this function will recursively partition until it finds the value that is in the nth position in the array. It should return this value to the calling program. This function will NOT completely sort the array.
- should be defined in recSorts.cpp/recSorts.h or recSorts.py or recSorts.cs