CS 2124 Data Structures

Programming Project 6

In this project, we will be implementing the HeapSort algorithm which uses a heap (priority queue) to sort an array of integers.

**Files provided in the project:**

1. **abc123Project6.c**. This is the only source code file in this project. It contains a main() function that is completed for you. It creates an array of size 200,000 randomly chosen integers (random number generator not seeded so you get the same results each time which can be helpful for debugging). It then calls a heapSort function (which currently is empty) and then prints out the contents of the array (which should be sorted at the end of your project). It contains the following function prototypes. You need to implement the functions in this file below main:
   1. **void heapSort(int \*arrayToSort, int n).**

heapSort takes arrayToSort, an int pointer to an array that we need to sort, and n, the size of the array, and puts the elements of arrayToSort into non-decreasing order using a heap to repeatedly find the largest element that we have not yet placed into the "sorted portion" of keyM.

* 1. **int leftChild(int i);**

Given index i, return the index of the left child of node i.

* 1. **int rightChild(int i);**

Given index i, return the index of the right child of node i.

* 1. **int parent(int i);**

Given index i, return the index of the parent of node i.

* 1. **void makeHeap(int \*heap, int n).**

Given an array of integers, makeHeap arranges the elements int a valid heap configuration where each number is greater than both of its children.

* 1. **void heapifyUp(int \* heap, int i).**

heapifyUp is used to restore the heap to a valid configuration after inserting a new value at the end of the heap. i is the index of the new value. While it is larger than its parent, swap the value with the parent (or stop when it becomes the root and does not have a parent).

* 1. **void heapifyDown(int \*heap, int n, int i).**

heapifyDown is used to restore the heap to a valid configuration after removing the largest value at the top of the heap. After removing the maximum element at the top of the heap, we would have swapped a value from the bottom of the heap into the top position. This value may be smaller than some of its children. If so, we need to swap this value with the larger of its two children. Stop when it is larger than all its children (or when it no longer has any children).

2) **Makefile**. Update the makefile to reflect your abc123. Compile using **make p6** Execute the program using **./p6**

**Submitting:**

Zip up your project folder in a file name abc123Project6.zip (with your own abc123) and submit on the dropbox on Blackboard by the due date.

**Extra Credit:**

Implement the Selection Sort algorithm as we described in class and call it on an array of the same size and output the execution time of Selection Sort and HeapSort.