Lab 6

CSC 172 (Data Structures and Algorithms)
Fall 2018
University of Rochester
Due Date: Sunday, Nov 11 2018 @ 11:59 PM

Problem 1: Heap

1. (30 points) Create a new class `Heap` This class must implement the following two methods:
   (a) `heapify` which takes an array of integers as input and converts it into a max heap.
   (b) `heapsort` which takes an array and sorts it (ascending order)

   For heapsort you should call `heapify` method internally.

2. (30 points) Create another class `HeapTest` for testing. You must follow the following steps:
   (a) Create an array `arr` containing the following elements:
   5, 18, 3, 25, 27, 45, 97, 88, 26, 16, 49, 67
   (b) Call `heapify(arr)`
   (c) Print elements in arr
   (d) Create another array `arr2` containing the following elements:
   15, 99, 3, 77, 27, 45, 7, 88, 26, 5
   (e) call `heapsort(arr2)`
   (f) Print elements in `arr2`

Note: For getting full credits for part 1, your part 2 must work correctly.

Problem 2: Radix Sort

Radix Sort is typically implemented to support only a radix that is a power of two. This allows for a direct conversion from the radix to some number of bits in an integer key value. For example, if the radix is 16, then a 32-bit key will be processed in 8 steps of 4 bits each. Re-implement (in Java) the Radix Sort implementation of the lecture to use bit shifting in place of division. Use at least 5 test cases to compare the running time of the old and new Radix Sort implementations. Submit the Java code and your findings in a separate text file.

Submission

Create a zip file Lab6.zip containing your source code for Problem 1 and 2 and a README file. Submit this file at the appropriate location on the Blackboard system at `learn.rochester.edu`. The README file should state your and your team member's name and any other pertinent information. You should include the following Java files:

- Heap.java
- HeapTest.java
- RadixSort.java

Grading **(100 pts)**

Problem 1: 60 pts
Problem 2: 40 pts