COS 211—Fundamental Data Structures
Time heaps built top-down and bottom-up, and time heapsort using a large `int` array
Practical 4 Term 2 9 May 2016

Submit before: 15 May 2016 : 23h59

Submit as soon as possible using the `cd; make submit` commands. Put today's code for minimizing heaps into your `42practical` directory.

---

1. As usual, inside your namefile directory, create another directory called `42practical` by copying an old correctly working directory into it as in the example below.

```
1  cd smith,william
2  cp -r 32practical/ 42practical
```

   Next fix the `Makefile` so that today's prac can be run by typing only the `make` command at the command line interface. If this does not work, your submission will be marked down with 10 marks. Edit the `Makefile` in `42practical` so that the name of the object, `OBJ`, corresponds with the name of your java main program.

2. Data has been provided as five lines in the file `heap.data` in the `../ds/notes` directory.

```
12 3 5 7 10 11 23 19 29 17 31 41
6 18 17 2 1 9 14 12 3 5 7 10 11 23 15
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5
22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
5064877 8699311 3330659 17806681 18003133 12198083 10673129 ... very long line
```

   This data is provided as four short lines of `heap.data`. The fifth line contains the large data set.

3. Please use one main program with two phases—a **testing** phase and a **timing** phase. In the test phase

   Test the code using four the small data sets.
   Use the four lines of data provided to test your code before running the timing tests.

   Time the code with the large set of data.
   When you have shown that your code works correctly for the four small data sets you may then proceed to do the timing on the largish shuffled array provided in `heap.data` in the usual location. There are are a million or so numbers in line five. Please read the data in place, i.e., from where it is stored in the file system,[1] and *do not copy it to your own working area*—there is a penalty of 10 marks for not complying with this instruction.

   Each part will (a) build the input array into a heap top down, (b) build the heap bottom up, and (c) sort the same data after inserting it into a heap bottom up.

4. Create a `Heap` class. Implement `private heapify(int i)` where `i` is the position of the node to be heapified. The class contains some procedures called:

[1] If you are using a laptop then copy the data to a file with the path `/export/home/notes/ds/heap.data`.

```
public void buildHeapTopDown(Node A []).
```

and

```
public void buildHeapUp(Node A []).
```

and a method that can be called to check if `H` is a heap

```
  H.isHeap();
```

This predicate returns a boolean that is true when `this` is a heap. Next, program heapsort.

```
public Node[] sort (Node Z[], int sizeZ).
```

The sort method is given an array `Z` and its size as arguments when it is called. Note that the size of `Z` must also be given because it can differ from the holder in which `Z` is stored. It copies `Z` to the array in the heap class and then sorts it and returns the sorted array as a value. The sorting is done from *largest to smallest*. Note that the array `Z` is NOT the private array inside the heap in which all the work is done. The data structure `Node` can be any convenient comparable type—the keys and the data may be conflated, call the type of the each entry in the heap class a `Node`, even if it is a simple integer.

Test your heap code by building a minimizing heap from each of the four lines of data provided in the file `heap.data`. We suggest that you use a `for` loop to treat only the first four lines of data, and make the loop run to five lines when you are certain that your code is working correctly.

5. Your code will be assessed on the output it gives for each of the five lines of input. It must print two heaps and the heap sorted result for each line of input as well as their respective timings. Separate each of the five data sets with a line of 50 hyphens. So, for each of the data sets print *only the first eight elements* of the heap and then the timing on a separate line. Print the first eight sorted numbers and the last eight sorted numbers on two lines and the timing of the sort on a separate line and follow all this with a line of 50 hyphens.
