Implementation of data structures and algorithms
Short Project 4: Priority Queues

Due: 11:59 PM, Feb 16, 2020.

Submission procedure: same as usual.

Team task:

1. Implement binary heap. Starter code (BinaryHeap.java) is provided.

Practice task (optional):

2. Implement heap sort and its related methods in the binary heap class. Write a driver that uses heap sort to sort an array in ascending order, and then in descending order.

3. Implement the problem of finding the kth largest element of a stream (or k largest elements) using the following algorithm: maintain the k largest elements seen so far in a priority queue (min heap for finding k largest elements). Implement replace() method in binary heap class. Compare its performance with code that uses Java's priority queue (use small values for k (100, say) and large values of n).

Use Java's PriorityQueue for the following questions.

4. Implement Huffman Coding algorithm. Create a class for representing coding trees. Use a priority queue to hold the trees. In each step, the algorithm removes two trees with the smallest frequencies, merges them, and inserts it back into the priority queue. At the end, there is a single coding tree. Traverse the tree and output the binary codes for each symbol.

5. Perfect powers: Write an algorithm to output exact powers (numbers of the form a^b, b>1), in the range 2 to n.

6. Given an array of prime numbers, output numbers in [2,n] all of whose prime factors are only from the given set of prime numbers. For example, given {3,7}, the program outputs {3,7,9,21,27,49,63,...}. Make sure that your program outputs each number only once, in sorted order.