Spring 2024 COP 3503 Exam #1 2/6/2024 – Part A (Java API) Solution

1) (10 pts) What is the expected run-time of each of the following operations in the Java API? For each of these questions, assume that the data structure in question currently has ***n*** items in it.

(a) Collections: sort(List<T> list) **O(nlgn)**

(b) ArrayList: add(E e) **O(1)**

(c) ArrayList:add(int index, E element) **O(n)**

(d) TreeSet: lower(E e) **O(lg n)**

(e) HashMap: get(Object key) **O(1)**

(f) PriorityQueue: poll() **O(lg n)**

(g) PriorityQueue: remove(Object o) **O(n)**

(h) ArrayList: contains(Object o) **O(n)**

(i) HashSet: add(Object o) **O(1)**

(j) ArrayList: size() **O(1)**

Grading: 1 pt all or nothing on each.

2) (15 pts) The rolling median problem is as follows: As you read in ***n*** integers, after reading in each one, output the current median of the values. (Recall, that the median of a list of an odd number of values is the middle value, and that the median of a list of an even number of values is the average of the two middle values.) For this question, you'll complete the code on the back of this page so that it prints out **twice the current median value**. (This is so we can avoid floating point numbers…)

The strategy to solve the problem efficiently is as follows: Keep two priority queues, one a max queue which will store the "smaller half" of values (called left in the code) and another priority queue (a min queue), which will store the "larger half" of values. After reading in each number, insert it into the appropriate side (for example, if the left has [6, 2, 9] and right has [15, 13, 22] and we're inserting 19, it will go on the right.) Then, in the worst case, one of the two priority queues will have 2 more items than the other. If this is the case, then remove one item from the larger sized queue and put it into the smaller sized queue. Once this is done, the median is easy to calculate. (We'll let you figure it out from here.)

The Java API methods you need will be described at the end of the next page.

import java.util.*;

```
public class rollingmedian {
    public static void main(String[] args) {

        Scanner stdin = new Scanner(System.in);
        int n = stdin.nextInt();

        PriorityQueue<Integer> left = new
            PriorityQueue<Integer>(Collections.reverseOrder());
        PriorityQueue<Integer> right = new PriorityQueue<Integer>();

        for (int i=0; i<n; i++) {

            int val = stdin.nextInt();

            if (i == 0) {
                left.offer(val);
                System.out.println(2*val);
                continue;
            }

            // Decide which side to add val to.
            if (val > left.peek())  // several ways, > or >= okay… // 1 pt
                right.offer(val);                                 // 1 pt
            else                                                  // 1 pt
                left.offer(val);                                  // 1 pt

            // Transfer value from left to right or right to left if necessary.

            if (left.size() > right.size()+1) right.offer(left.poll());// 2pts
            if (right.size() > left.size()+1) left.offer(right.poll());// 2pts

            // Print out twice the current median (3 cases)
            if (left.size() > right.size())                       // 1 pt
                System.out.println(2*left.peek());                // 1 pt
            else if (right.size() > left.size())                  // 1 pt
                System.out.println(2*right.peek());               // 1 pt
            else                                                  // 1 pt
                System.out.println(left.peek()+right.peek());     // 2 pts


        }
    }
}
```