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

Last Name: _____ First Name: _____

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) _____

(b) ArrayList: add(E e) _____

(c) ArrayList:add(int index, E element) _____

(d) TreeSet: lower(E e) _____

(e) HashMap: get(Object key) _____

(f) PriorityQueue: poll() _____

(g) PriorityQueue: remove(Object o) _____

(h) ArrayList: contains(Object o) _____

(i) HashSet: add(Object o) _____

(j) ArrayList: size() _____

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.









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









            // Print out twice the current median (3 cases)









        }
    }
}

// PQ: offer(E e) - adds e to this PQ
// PQ: poll() - removes and returns the head of this PQ
// PQ: peek() - returns but doesn't remove head of this PQ
// PQ: size() - returns the number of elements of this PQ
```

Spring 2024 COP 3503 Exam #1 2/6/2024 – Part B (Backtracking)

Last Name: _____ First Name: _____

3) (5 pts) The method to find the "Magic Sum" for the Hexagram program (Program 2) was to take the 12 input numbers, add them, and divide this sum by three. Why does this work?

4) (5 pts) In class, code to solve the Tentaizu problem was executed, which ran fairly quickly. A single line of code was commented out and then the solution ran quite slowly (it solved one puzzle in several seconds, while we waited.) Conceptually, what did that line of code do?

5) (5 pts) In class, we visualized a solution to the Eight Queens puzzle as a permutation of the integers from 1 to n, where the i^{th} integer in the permutation represented which row to place the queen in column i. Using this same visualization technique, draw the solution that corresponds to the first permutation lexicographically that correctly solves the five queens problem below.

6) (15 pts) A prefix composite number is one such that each prefix of it is also a composite number. For example, 44052 is a prefix composite number that is 5 digits long because 4, 44, 440, 4405 and 44052 are all composite numbers. Complete the program on below so that it prints out all prefix composite numbers that are 5 digits long.

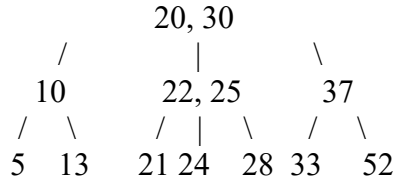
```
import java.util.*;

public class prefixcomposite {

     public static void main(String[] args) {
          go(0, 0, 5);
     }

     // cur is a valid prefix-composite of k digits
     // n is the # of digits in the prefix-composites to be printed
     public static void go(int cur, int k, int n) {









     }

     // Returns true if n is prime or if n is less than 2.
     // Must run in O(sqrt(n)) time and not use any doubles for
     // full credit.
     public static boolean isprime(int n) {









     }
}
```

Spring 2024 COP 3503 Exam #1 2/6/2024 – Part C (DJ Sets, 2-4 Trees)

Last Name: _____ First Name: _____

7) (5 pts) Draw a valid 2-4 Tree of height 1 with 15 distinct positive integers in it.

8) (5 pts) Show the result of deleting 13 from the 2-4 Tree shown below:



9) (15 pts) One application of a disjoint set is to mark the number of connected regions on a 2D grid. Consider an input grid of upper case letters. A connected region is one which consists of the same letter which share an edge on the grid (so a square that is up, down, left or right from another is connected to it. For example, in the grid below there are 5 connected regions (2 marked A, 1 marked B, C and D.)

A	A	D	B	A
A	A	D	B	B
A	D	D	C	C
A	A	D	D	C

Complete the code on the next page so that it reads in this grid and outputs the # of connected regions in the grid. **(Note: Disjoint Set Code at end of Part D.)**

```
import java.util.*;

public class connected {

    public static int r;
    public static int c;
    public static char[][] g;

    final public static int[] DX = {0,1};
    final public static int[] DY = {1,0};

    public static void main(String[] args) {

        Scanner stdin = new Scanner(System.in);
        r = stdin.nextInt();
        c = stdin.nextInt();
        g = new char[r][];
        for (int i=0; i<r; i++)
            g[i] = stdin.next().toCharArray();

        djset dj = new djset(r*c);

        for (int i=0; i<r; i++) {
            for (int j=0; j<c; j++) {
                for (int k=0; k<2; k++) {

                    /*** FILL IN CODE HERE ****/









                    // end k}
            } // end j
        } // end i

        System.out.println(dj.numSets);
    }
}
```

Spring 2024 COP 3503 Exam #1 2/6/2024 – Part D (Skip Lists, Red-Black Trees)

Last Name: _____ First Name: _____ Rec Time: ____

10) (4 pts) The default description of a skip list indicates that when inserting a value at level k, there is a 50% chance (1 in 2) that the same value should be inserted into level k+1. During class, students looked up some information online and suggested that I try two other fractions of the form 1/x and 1/y, where x was an integer and y was an irrational number. What are those two numbers?

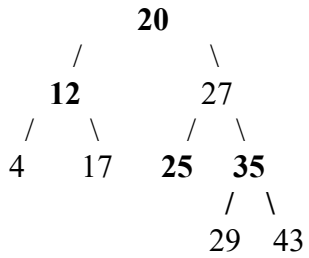
x = ____ y = ____

11) (2 pts) Of the values suggested above, which one performed the fastest on the large test run in class?

____

12) (3 pts) In most circumstances, the skip list insertion code in class follows the level k to level k+1 rule based on a randomly generated value. There's one exception to this circumstance, what is it?

13) (6 pts) Show the final result of deleting the value 25 from the red black tree depicted below. Black nodes are bolded and red nodes are not. (Note: There are two valid correct answers based on the typed notes from class.)



15) (5 pts) Instead of using oil to cook food, what does an air fryer use, heated by a powerful fan, to cook food?

_____

Disjoint Set Code for Part C Question 9 (it's mislabeled on the test as 5)

```
class djset {

      public int[] par;
      public int numSets;

      public djset(int n) {
            par = new int[n];
            for (int i=0; i<n; i++) par[i] = i;
            numSets = n;
      }

      public int find(int u) {
            if (par[u] == u) return u;
            return par[u] = find(par[u]);
      }

      public boolean union(int u, int v) {
            u = find(u);
            v = find(v);
            if (u == v) return false;
            numSets--;
            par[v] = u;
            return true;
      }
}
```