COP 3503 Exam 1 Review Questions for Fall 2026

Part A (Java API) – From Spr 24 Exam 1

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)









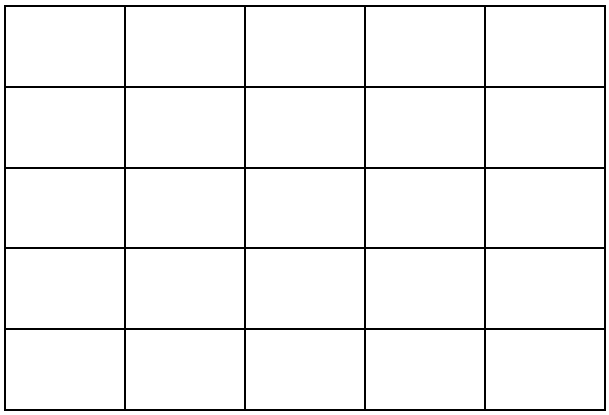
        }
    }
}

// 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
```

Backtracking – Spr 24 Exam (except question 3)

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) {









     }
}
```

Greedy Algorithms – Spr 24 Exam 2

4) (10 pts) A Huffman tree has been created for a particular input file comprised solely of the characters ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’ with the codes listed:

Letter	Huffman Code	Possible Letter Frequency
‘a’	1111	
‘b’	10	
‘c’	11101	
‘d’	0	
‘e’	111001	
‘f’	110	
‘g’	111000	

(a) Draw the corresponding Huffman Tree below.

(b) Fill in possible letter frequencies, ***without any tie cases***, for each of the seven letters in both the tree and the table above that could create this set of Huffman codes. (Note: Both the frequencies of each letter must be distinct AND when comparing items in the PQ, there should never be a tie between the second and third smallest items in it.

5) (15 pts) Arup's World Wide Coffee Emporium sells cups of coffee. He sells several different types of coffee. Each type of coffee has the following attributes: cost (# of cents it costs Arup to make one cup of this type of coffee), sell (# of cents Arup sell's each cup of this type of coffee), and numcups (the number of cups he has in inventory of this type of coffee.) For this question: write the compareTo method for the coffeecup class AND write a function which takes in an array of type coffeecup storing Arup's total inventory and an integer, totalcups, and returns the maximum profit he can make by selling exactly totalcups number of cups of coffee.

```
class coffeecup implements Comparable<coffeecup> {
     public int cost;
     public int sell;
     public int numcups;

     // Constructor omitted.

     public int compareTo(coffeecup other) {









     }
}
public static long maxProfit(coffeecup[] list, int totalcups) {









}
```

Sorting, Alg Analysis – Spr 24 Exam 2

2) (8 pts) Show the result of each iteration of a Radix Sort on the following values. The last column has been filled in.

Initial List	First Sort	Second Sort	Third Sort	Fourth Sort	Sorted List
32361					24169
48147					32361
32661					32647
58347					32661
34629					34629
24169					48147
48341					48341
32647					58347

3) (11 pts) Let T(n) represent the average number of nodes with two children in a binary search tree. Assuming that the root node of a binary search tree has probability of $\frac{1}{n}$ of being any particular rank (from 1 to n) in the sorted list of values stored in the tree, write down a recurrence relation that T(n) satisfies **and simplify this recurrence so that there's a single sum added to a single term on its right-hand side**.