Name	UIN	Seat

Question:	1	2	3	4	Total
Points:	10	10	10	10	30
Score:					

CS 125 Practice Final Exam

Please fill out your name and UIN. Also write your UIN at the bottom of each page of the exam in case the pages become separated.

Practice exam questions are intended to represent the types and difficulty level of questions you should expect to encounter on the real final exam. However, since one of the main goals of the practice exam is to give you practice writing code using pen and paper, questions that require code writing may be overrepresented.

Answer each question as **clearly** and **succinctly** as possible. Draw pictures or diagrams if they help. The point value assigned to each question may help you allocate your time. **No aids of any kind are permitted.**

I have neither given nor received help on this exam.

Sign and Date: _____

1. (10 points) Modeling an Airport

Write a Java class to represent an airport. You should model the aspects of airports that are important to planes as they land, drop off and pick up new passengers, and take off again.

Specifically, your airport class should model the following airport *properties* (2 points):

- a unique identifier: "CMI" for the University of Illinois Willard Airport[1]
- location: CMI is located at 40°02′21″N 88°16′41″W
- some number of runways: CMI has three runways
- some number of gates: CMI has three gates

You can assume that the number of runways and gates does not change once the airport is created.

You should also model the following airport *operations* (6 points):

- **Landing:** a plane should be able to determine if it can land at a specific airport, and on which runway. Note that a plane should only be able to land if there is both a runway and gate available.
- **Parking:** once a plane lands it should be directed to a gate. You should ensure that two planes do not attempt to park at the same gate—this tends to annoy passengers and pilots.
- **Departing:** at some point a plane will request to leave its gate and take off via a runway. You should not allocate the same runway for a simultaneous takeoff and landing! Once plane has successfully cleared the airport it should notify your airport that it has departed and the runway is now free.

You can assume that a `Plane` class exists and has implemented `equals` correctly, allowing you to uniquely identify each plane that uses your airport.

Finally, your class should provide a way to look up an airport by its identifier (2 points). As you provide this feature, you can assume that no two airports are ever created with the same identifier, and that no more than some number of airports will ever be created.

[1] Fun fact: the University of Illinois is the only university that operates an airport.

Answer for Question 1

2. (10 points) Debugging

The following function contains bugs. Identify the bugs (5 points) using the existing code snippet, and then rewrite the code so that it is correct (5 points).

```
1  /**
2   * Count duplicate strings in an array.
3   *
4   * A string is a duplicate if it equals any other string in the array.
5   * For example:
6   *   countDuplicates({ "a", "b" }) should return 0
7   *   countDuplicates({ "a" }) should return 0
8   *   countDuplicates({ "a", "a" }) should return 2
9   *   countDuplicates({ "a", "b", "a", "a" }) should return 3
10  */
11 public long countDuplicates(final String[] strings) {
12   int count = 0;
13   for (int i = 0; i < strings.length; i++) {
14     for (int j = 0; j < strings.length; j++) {
15       if (strings[i] == strings[j]) {
16         count++;
17       }
18     }
19   }
20   return count;
21 }
```

Answer for Question 2

3. (10 points) Tree Recursion

Recall the binary tree structure that we used on MP6:

```
1  public class Tree {
2
3      /* Current node's parent. May be null if I'm the root of the tree. */
4      private Tree parent;
5
6      /* Current node's left child, or null if none. */
7      private Tree left;
8
9      /* Current node's left child, or null if none. */
10     private Tree right;
11
12     /* Current node's value. */
13     private int value;
14 }
```

And recall that when we insert a new element into the tree, we put it:

- *to the left* if the new value is less than the node we are comparing against, and
- *to the right* if the new value is greater than the node that we are comparing against.

Previously we did not consider how to handle duplicate elements[2]. First, determine how to extend our binary tree to handle duplicate elements (4 points) You will either need to modify the tree node definition, or the insertion algorithm.

Next, write a recursive algorithm that, given the root of the tree, counts the total number of duplicate elements (6 points)—similar to the previous question. This algorithm must work on the tree that can handle duplicates that you designed above.

[2] I realize that duplicates are becoming a theme of this practice exam.

Answer for Question 3

4. (10 points) Quicksort Pivoting

First, provide a brief high-level overview of the quicksort algorithm (2 points). Feel free to use pseudocode if this is easier for you.

Second, explain why the choice of pivot value is so important to quicksort. Using a diagram and a small example, describe what occurs in the best-case selection of pivot value (4 points), and also in the worst case (4 points). Make sure to describe the $\mathcal{O}(n)$ runtime in both cases.

Answer for Question 4