Name: _____

Computer Science I — Fall 2023

Final Exam

This exam should have six pages. Closed book and notes.
No calculators or computers allowed.

Problem 1: [10 points]

Below, define a static method called `longestString` that takes three Strings as inputs and returns the longest of the three strings. In the case of a tie, it doesn't matter which of the longest strings you return.

Problem 2: [20 points]

The questions below refer to the following mysterious method:

```
public static void mystery(int x, ArrayList<Integer> y) {
    int i=0;
    while(x > y.get(i)) {
        i=i+1;
    }
    y.add(i, x);
}
```

a) Describe *briefly*, in plain English, what the `mystery` method does. What would be a good, descriptive name for this method?

b) Describe some possible inputs you would use when calling the method if you wanted to test it thoroughly. Explain briefly why you're using each.

Problem 3: [20 points]

a) If Binary Search is so great, why would anyone ever use Linear Search? Explain.

b) We've seen methods that print information to the terminal window, and also methods that return values. What's an advantage of returning a value instead of printing it?

c) For an array of a given size, does it take more computational work to search the array or to sort it? Explain.

d) Why is it better to write lots of small unit tests, instead of testing the program as a whole?

Problem 4: [20 points]

I've been commissioned to write a class called `Dice` that keeps track of a *group* of Die instances, and has methods that operate on the whole collection. I'm too busy grading final exams, so I need your help with a couple of the methods. The class uses an *array* to hold the Die instances, as shown in the start of the class below:

```
public class Dice
{
    private Die[] dice;  // To hold all the Die instances

    // Methods omitted
}
```

a) Please define the *constructor* for the `Dice` class. It should take two `int`s as inputs: The number of `Die` instances desired, and the number of sides each die should have (all `Die` instances in the group will have the same number sides). It should create the specified number of `Die` objects and store them in the array (after creating the array, of course).

b) Define a method called `totalRoll`. It should roll each `Die` in the group once, and return the *sum* of the rolled values.

Problem 5: [20 points]

Finish the definition of the method below. It takes the name of a file as its input, and should read all of the integers in the specified file and return the largest. You may assume that if the file exists, it contains nothing but integers, and that there's at least one integer in the file. (There may be more than one per line, however, separated by spaces.) For full credit, you should return -1 if the file does not exist. You may assume that all appropriate `import` statements are at the top of the class, and therefore you do not need to write any yourself.

```
public int largestValueInFile(String filename)
                              throws FileNotFoundException {
```

Problem 6: [10 points]

The binary search method we wrote in class (and that you worked with in lab) is shown below, with some portions omitted. Fill in the blanks with the appropriate missing code.

```
public boolean binarySearch(int[] data, int key) {
     int left;
     int right;
     left = _________;
     right = _________;

     while (____________________) {
          int mid = (right+left)/2;
          if (key < data[mid]) {
               ____________________;
          }
          else {
               ____________________;
          }
     }
     return (data[left] == key);
}
```