CSE142 Sample Final Exam
Autumn 2018

1. Reference Mystery, 5 points. The following program produces 5 lines of output. Write the output below, exactly as it would appear on the console.

```
import java.util.*;
public class ReferenceMystery {
    public static void main(String[] args) {
        int x = 1;
        int y = 7;
        int[] data = {2, 4, 6};
        System.out.println(x + " " + y + " " + Arrays.toString(data));

        mystery1(x, y);
        System.out.println(x + " " + y);

        x = mystery2(data[x], data);
        System.out.println(x + " " + y + " " + Arrays.toString(data));

    }

    public static void mystery1(int x, int y) {
        y++;
        x = x / y;
        System.out.println(x + " " + y);
    }

    public static int mystery2(int z, int[] numbers) {
        z = z * 4;
        numbers[z % 2]++;
        z = -1;
        numbers[z + 1]--;
        System.out.println(z + " " + Arrays.toString(numbers));
        return 5;
    }
}
```

2. Array Simulation, 10 points. You are to simulate the execution of a method that manipulates an array of integers. Consider the following method:

```
    public static void mystery(int[] list) {
        for (int i = 1; i < list.length; i++) {
            if (list[i - 1] % 2 == 0) {
                list[i - 1]++;
                list[i]++;
            }
        }
    }
```

In the left-hand column below are specific arrays of integers. You are to indicate in the right-hand column what values would be stored in the array after method mystery executes if the integer array in the left-hand column is passed as a parameter to mystery.

Original Array	Final Array
[10, 10, 10]	____________________________
[2, 3, 4, 5]	____________________________
[3, 4, 5, 7, 9]	____________________________
[2, 3, 5, 7, 9]	____________________________
[4]	____________________________

3. Inheritance Mystery, 6 points. Consider the following classes:

Given the classes to the left, write the output produced by the client code exactly as it would appear on the console.

```
public class Stew extends Soup {
   public void method2() {
      System.out.print("stew 2    ");
      method1();
   }
}

public class Soup {
   public void method1() {
      System.out.print("soup 1    ");
   }

   public void method2() {
      System.out.print("soup 2    ");
   }

   public String toString() {
      return "soup";
   }
}

public class Chowder extends Soup {
   public void method1() {
      method2();
      System.out.print("chowder 1    ");
   }

   public String toString() {
      return "chowder";
   }
}

public class Chili extends Stew {
   public void method1() {
      System.out.print("chili 1    ");
   }

   public String toString() {
      return "chili";
   }
}

// client code
public static void main(String[] args) {
   Soup[] bowl = { new Stew(), new Soup(),
                   new Chowder(), new Chili() };

   for (int i = 0; i < bowl.length; i++) {
      System.out.println(bowl[i]);
      bowl[i].method1();
      System.out.println();
      bowl[i].method2();
      System.out.println();
      System.out.println();
   }
}
```

4. File Processing, 9 points. Write a static method named countCoins that accepts as its parameter a Scanner for an input file whose data represents a person's money grouped into stacks of coins. Your method should add up the cash values of all the coins and print the total money at the end. The input consists of a series of pairs of tokens, where each pair begins with an integer and is followed by the type of coin, which will be either "pennies" (1 cent each), "nickels" (5 cents each), "dimes" (10 cents each), or "quarters" (25 cents each), case-insensitively. A given coin might appear more than once on the same line.

   For example, suppose the input file contains the following text:

```
3 pennies 2 quarters 1 pennies 3 nickels 4 dimes
```

   Three pennies are worth 3 cents, 2 quarters are worth 50 cents, 1 penny is worth 1 cent, 3 nickels are worth 15 cents, and 4 dimes are worth 40 cents. The total of these is 1 dollar and 9 cents, therefore your method would produce the following output if passed this input data. Notice that it says 09 for 9 cents.

```
Total money: $1.09
```

   Here is a second example. Suppose the input file contains the following text (notice the capitalization and spacing):

```
12    QUARTERS       1    Pennies            33
PeNnIeS

10
niCKELs
```

   Then your method would produce the following output:

```
    Total money: $3.84
```

   You may assume that the file contains at least one pair of tokens. You may also assume that the input is valid; that the input has an even number of tokens, that every other token is a positive integer, that the remaining tokens are valid coin types.

5. File Processing, 10 points. Write a static method gradeQuiz that takes as its parameter a Scanner containing the questions and correct answers for a quiz along with a particular student's responses. Your method should determine whether or not the student correctly answered each question and then print out the student's final score.

Each question in the input file will be represented by three lines. The first line will contain the question, the second line will contain the correct answer, and the third line will contain the student's response. A student's response is considered correct if it matches the correct answer, case-insensitively.

For example, suppose the input contains the following text:

```
Who is the CSE 142 instructor?
Brett Wortzman
Brett
What language is taught in CSE 142?
Java
JAVA
What is the best drink in the world?
Coffee
coffee
Is a poptart a sandwich?
No
NO
Is an oreo a sandwich?
No
yes
```

This quiz contained 5 questions, and the student got the second, third, and fourth questions correct. In this case, your method would produce the following output:

```
Question 1: Who is the CSE 142 instructor?
Student's response: Brett
Correct answer: Brett Wortzman
Incorrect. :-(

Question 2: What language is taught in CSE 142?
Student's response: JAVA
Correct answer: Java
Correct!

Question 3: What is the best drink in the world?
Student's response: coffee
Correct answer: Coffee
Correct!

Question 4: Is a poptart a sandwich?
Student's response: NO
Correct answer: No
Correct!

Question 5: Is an oreo a sandwich?
Student's response: yes
Correct answer: No
Incorrect. :-(

3/5 questions answered correctly.
```

You may assume that the input file contains text representing at least one question, that the number of lines in the input file is a multiple of three, and that the file has the format specified above. You should output the question, correct answer, and student response exactly as they appear in the input file.

6. Arrays, 10 points. Write a static method named range that accepts an array of ints as a parameter and returns the range of values in the array. The range is defined as the difference between the largest value and the smallest value.

   For example, if a variable called arr stores the following array:

```
[1, 4, 2, -1, 8]
```

   the call range(arr) would return 9 (the difference of 8, the largest value and -1, the smallest value).

   You may assume the given array is not null and that it contains at least one element. Your method must not modify the contents of the array.

7. Classes, 10 points. Suppose that you are provided with a pre-written class ClockTime as described below. Assume that the fields, constructor, and methods shown are implemented. You may refer to them or use them in solving this problem.

```
// A ClockTime object represents an hour:minute time during
// the day or night, such as 10:45 AM or 6:27 PM.
public class ClockTime {
    private int hour;
    private int minute;
    private String amPm;

    // Constructs a new time for the given hour/minute
    public ClockTime(int h, int m, String ap) { /* implementation omitted */ }

    // returns the field values
    public int getHour() { /* implementation omitted */ }
    public int getMinute() { /* implementation omitted */ }
    public String getAmPm() { /* implementation omitted */ }

    // returns String for time; for example: "6:27 PM"
    public String toString() { /* implementation omitted */ }

    // advances this ClockTime by the given # of minutes
    public void advance(int m) { /* implementation omitted */ }

    // your method will go here
}
```

   Write an instance method named isLaterThan that will be placed inside the ClockTime class. The isLaterThan method takes another ClockTime object as a parameter and returns true if this ClockTime represents a time later in the day than the ClockTime object passed in. If the object passed is later in the day, or if the two times are the same, the method returns false.

   For example, suppose the following objects are declared in client code:

```
ClockTime t1 = new ClockTime(8, 30, "AM");
ClockTime t2 = new ClockTime(1, 15, "PM");
ClockTime t3 = new ClockTime(9, 30, "AM");
ClockTime t4 = new ClockTime(9, 0, "AM");
ClockTime t5 = new ClockTime(8, 40, "AM");
ClockTime noon = new ClockTime(12, 0, "PM");
ClockTime mid = new ClockTime(12, 0, "AM");
```

The table below indicates the return value for various calls to isLaterThan:

Call	Return Value
t1.isLaterThan(t2)	false
t1.isLaterThan(t3)	false
t1.isLaterThan(t4)	false
t1.isLaterThan(t5)	false
t2.isLaterThan(t1)	true
t2.isLaterThan(t4)	true
t3.isLaterThan(t4)	true
t1.isLaterThan(noon)	false
t1.isLaterThan(mid)	true
t2.isLaterThan(noon)	true
t2.isLaterThan(mid)	true
noon.isLaterThan(mid)	true
t1.isLaterThan(t1)	false

Your method should not modify the state of either ClockTime object. You may assume that the state of both ClockTime objects is valid at the start of the call, and that both amPm fields store either "AM" or "PM".

8. Critters, 15 points. Write a class named Wombat that extends the Critter class from homework 8. Write the complete class with any fields, constructors, etc. that are necessary. All unspecified aspects of Wombats use the default behavior.

   Wombats burrow by moving in a southward pattern, going north every third move to get dirt out of their way. Their overall movement pattern is S, S, N, S, S, N, S, S, N, S, S, N, etc.

   A Wombat roars when it gets into a fight with something that looks like an Ant (displayed as a "%"). When fighting any other kind of Critter, a Wombat pounces. After winning a fight, a Wombat takes on the appearance of its defeated opponent. For example, if a Wombat fights an Ant and wins, it will then be displayed as "%" until it fights again. When Wombats are first created and haven’t fought yet, they are displayed as "?".

   When a Wombat encounters food, it randomly chooses whether or not to eat. Each time a Wombat encounters food, it should have a 2/3 chance of not eating and a 1/3 chance of eating, regardless of whether or not it previously ate.

9. Arrays, 15 points. Write a static method named compressTriplets that accepts an array of integers as a parameter and returns a new array of integers containing the sums of each successive set of three integers in the original array. If the given array has a length that is not a multiple of 3, the last sum in the returned array should be that of the remaining values.

   Below are example arrays and the expected returned array after calling compressTriplets:

contents of int[] arr	result of compressTriplets(arr)
[0, 3, 1, 2, 4, 3]	[4, 9]
[0, 3, 1, 2, 4, 3, -8]	[4, 9, -8]
[0, 3, 1, 2, 4, 3, -8, 3]	[4, 9, -5]
[1]	[1]
[]	[]

   You may assume the given array is not null. Your method must not modify the contents of the original array and must not construct any data structures except for a single integer array.

10. Programming, 10 points. Write a static method called count that takes as parameters a target string and a source string and that returns a count of the number of occurrences of the target string in the source string, regardless of casing. For example, the call:

```
count("i", "Mississippi")
```

would return 4 because there are 4 occurrences of the string "i" in the string "Mississippi". The call:

```
count("iss", "MISSISSIPPI")
```

would return 2 because there are two occurrences of "iss" in "MISSISSIPPI".

Your method should consider all possible starting positions for the target string. For example, the call:

```
count("EE", "EeEeE")
```

would return the value 4 because there are 4 different locations where the string "ee" occurs in the string "EeEeE" (starting at index 0, index 1, index 2, and index 3).

You may assume that the both the source and target strings are not empty.