CSE142 Sample Midterm
Summer 2019

1. Expressions, 10 points. For each expression in the left-hand column, indicate its value in the right-hand column. Be sure to list a constant of appropriate type (e.g., 7.0 rather than 7 for a double, Strings in quotes, true and false for booleans).

Expression	Value
`1 * 2 * 3 + (4 - 5)`	_______________
`28 % 4 + 18 % 5 % 5 + 9`	_______________
`1000 * 2 + 18 / 2 / 2 * 2`	_______________
`1 / 10.0 + "1" + 17 * 2`	_______________
`0.25 * 2 - 0.5 + 1 / 2`	_______________

2. Parameter Mystery, 12 points. Consider the following program.

```
public class ParameterMystery {
    public static void main(String[] args) {
       String green = "i";
       String am = "green";
       String ham = "sam";
       String i = "eggs";
       String eggs = "am";
       String sam = "ham";

       mystery(sam, i, am);
       mystery(ham, green, eggs);
       mystery(green, eggs, ham);
       mystery(i, sam, "green");
    }

    public static void mystery(String eggs, String ham, String green) {
       System.out.println("I do not like " + green + " " + eggs + " and " + ham);
    }
 }
```

Write the output produced by this program in the box below.

```
 ___________________________________________________________
|                                                           |
|                                                           |
|                                                           |
|                                                           |
|                                                           |
|                                                           |
|                                                           |
|                                                           |
|                                                           |
|                                                           |
|                                                           |
|                                                           |
|___________________________________________________________|
```

3. If/Else Simulation, 12 points. Consider the following method.

```
public static void ifElseMystery(int one, int two) {
    if (one % two == 0) {
        one = one / two;
    }
    if (two > one) {
        two--;
    } else if (two == one) {
        one = one + 5;
    }

    System.out.println(one + " " + two);
}
```

For each call below, indicate what output is produced.

```
  Method Call                       Output Produced
-----------------------------------------------------
  ifElseMystery(12, 45);           _______________

  ifElseMystery(15, 5);            _______________

  ifElseMystery(64, 8);            _______________

  ifElseMystery(12, 12);           _______________

  ifElseMystery(20, 7);            _______________

  ifElseMystery(100, 5);           _______________
```

4. While Loop Simulation, 12 points. Consider the following method:

```
public static void whileMystery(int n) {
    int x = 1;
    int y = 1;
    while (n > 2) {
        x = x + y;
        y = x - y;
        n--;
        System.out.print(y + " ");
    }
    System.out.println(x);
}
```

For each call below, indicate what output is produced:

```
  Method              Output Produced
-----------------------------------------------
  mystery(5);         _______________

  mystery(3);         _______________

  mystery(7);         _______________

  mystery(0);         _______________
```

5. Assertions, 15 points. You will identify various assertions as being either

always true, never true or sometimes true/sometimes false at various points in program execution. The comments in the method below indicate the points of interest.

```
public static int mystery(int x) {
    int y = 1;
    int z = 0;
    // Point A
    while (x > y) {
        // Point B
        z = z + x - y;
        x = x / 2;
        // Point C
        y = y * 2;
        // Point D
    }
    // Point E
    return z;
}
```

Fill in the table below with the words ALWAYS, NEVER or SOMETIMES.

	x > y	z > 0	y % 2 == 0
Point A			
Point B			
Point C			
Point D			
Point E			

6. Programming, 15 points. Write a static method called testFairCoin that takes a console Scanner as a parameter and prompts the user for a series of coin flips. The user will input one of three words: "heads" if they flip a heads, "tails" if they flip a tails, or "done" to stop entering flips. Your method should compute and output the percentage of times the user flipped heads, and return whether or not this sequence of flips seems to represent a fair coin. A coin is considered to be fair if the user inputs "heads" between 45 to 55 percent of the time (inclusive).

   For example, suppose the following method calls were made:

```
Scanner console = new Scanner(System.in);
boolean result1 = testFairCoin(console);
System.out.println();
boolean result2 = testFairCoin(console);
```

   These calls would produce interactions like the following (user input bold and underlined):

```
next flip? heads
next flip? heads
next flip? tails
next flip? tails
next flip? done
was heads 50.0% of the time

next flip? tails
next flip? heads
next flip? tails
next flip? done
was heads 33.333333333333% of the time
```

   In the first log, the user enters four flips: two of which are heads and two of which are tails. Then they enter "done" to stop entering flips. In the second log, the user enters three flips before quitting: one of which is heads and two of which are tails. After this code is executed, the variable result1 (from the first log) is set to true because the first series of coin flips was heads 50 percent of the time, which is between the inclusive bounds of 45 to 55 percent. The variable result2 (from the second log) would be set to false because the percentage of heads was not between 45 and 55. Notice that the coin flips are being entered by the user, not generated by the method; you should not use a Random object to solve this problem.

   You may assume that the user always enters at least one flip, and does not enter any input other than the three words described ("heads" or "tails" or "done"). You should not round the percentage of heads computed. You must exactly reproduce the format of these logs, though the values may be different based on user input.

   Write your solution to problem 6 on the next page.

Write your solution to problem 6 here:

7. Programming, 15 points. Write a static method called busyDay that takes an integer num and a Random object as parameters and schedules a series of random meetings. Your method should repeatedly generate meetings between 15 and 60 minutes long (inclusive) until num meetings have been scheduled. After each meeting is generated, your method should print out the length of the new meeting and the total amount of meeting time scheduled so far. After all meetings have been scheduled, your method should print out the length of the longest scheduled meeting.

   For example, suppose the following calls were made:

```
Random rand = new Random();
busyDay(6, rand);
```

   These calls would produce output like the following:

```
Scheduled new 37-min meeting; total time now 0h 37m
Scheduled new 16-min meeting; total time now 0h 53m
Scheduled new 22-min meeting; total time now 1h 15m
Scheduled new 27-min meeting; total time now 1h 42m
Scheduled new 35-min meeting; total time now 2h 17m
Scheduled new 48-min meeting; total time now 3h 5m
Longest meeting was 48 minutes
```

   Suppose the following subsequent call is then made:

```
busyDay(3, rand);
```

   This call would produce output like the following:

```
Scheduled new 30-min meeting; total time now 0h 30m
Scheduled new 60-min meeting; total time now 1h 30m
Scheduled new 17-min meeting; total time now 1h 47m
Longest meeting was 60 minutes
```

   Notice that the total meeting time is given in the format Xh Ym. You are not required to add extra zeroes if either the total hours or minutes are a single digit.

   You may assume that at least one meeting will be scheduled. You must exactly reproduce the format of these logs, though the values may be different due to randomness.

   Write your solution to problem 7 on the next page.

Write your solution to problem 7 here:

8. Programming, 9 points. Write a method called isMonotonic that takes in an integer n and a boolean incr as parameters. If incr is true, your method should return true if the digits in n are strictly increasing. If incr is false, your method should return true if the digits in n are strictly decreasing. In all other cases, your method should return false. If two adjacent digits are equal, the number is not considered to be either strictly increasing or strictly decreasing.

Below are some sample calls to isMonotonic and their return values.

Method Call	Return Value	Method Call	Return Value
isMonotonic(0, true)	true	isMonotonic(8, false)	true
isMonotonic(11, true)	false	isMonotonic(12, false)	false
isMonotonic(21, true)	false	isMonotonic(22, false)	false
isMonotonic(29, true)	true	isMonotonic(642, false)	true
isMonotonic(1234, true)	true	isMonotonic(89, false)	false
isMonotonic(890, true)	false	isMonotonic(987, false)	true
isMonotonic(588, true)	false	isMonotonic(411, false)	false

You may assume that n is not negative. You may not use Strings or other objects to solve this problem.

Write your solution to problem 8 on the next page.

Write your solution to problem 8 here: