CSE142 Sample Midterm
Spring 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 + 6) / 7 + 8 % 9`	_____
`20 / 7 * 2.0 + 5.0 / 2 - (1 / 4)`	_____
`3 * 4 + "3" + 9 * 5 + 6`	_____
`(3 * 5 < 1 + 2) || (6 != 5 && !(7 < 7))`	_____
`1 + 1 * (1 - 1) + (1 + 1 + 1) % (1 + 1)`	_____

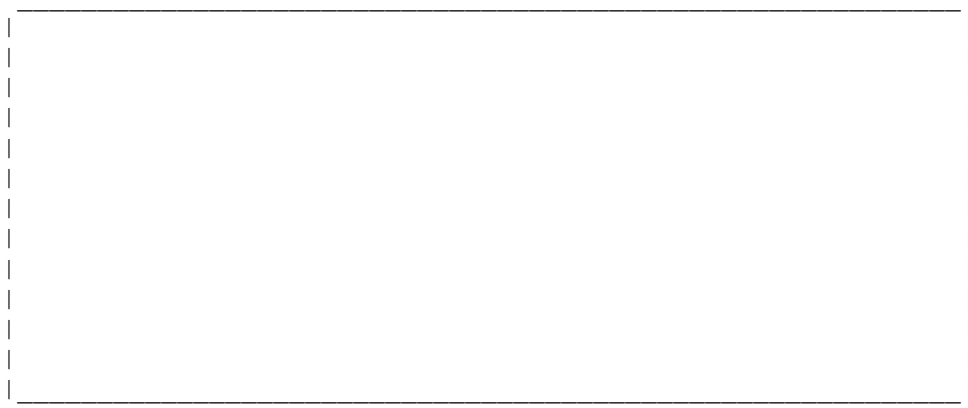
2. Parameter Mystery, 12 points. Consider the following program.

```
public class Mystery {
    public static void main(String[] args) {
        String snow = "sleet";
        String day = "slippery";
        String storm = "snow";
        String slippery = "storm";
        String weather = snow + storm;

        weather(snow, storm, day);
        weather(weather, slippery, storm);
        weather(storm + "storm", "snow" + snow, weather);
        weather = "sun";
        weather("sunny", weather, slippery);
    }

    public static void weather(String snow, String storm, String day) {
        System.out.println("a " + storm + " and " + day + " for " + snow);
    }
}
```

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 a, int b) {
    if (a == b || b > 2) {
        a++;
        b = b + 3;
    } else {
        b++;
    }
    if (a < b && a % 2 == 1) {
        a++;
        b = a - 2;
    } else if (b % 2 == 0) {
        a = a - 2;
        b = a + 3;
    }
    System.out.println(a + " " + b);
}
```

For each call below, indicate what output is produced.

Method Call	Output Produced
ifElseMystery(2, 2);	_____
ifElseMystery(3, 1);	_____
ifElseMystery(4, 0);	_____
ifElseMystery(5, 3);	_____
ifElseMystery(1, 2);	_____
ifElseMystery(7, 4);	_____

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

```
public static void mystery(int y) {
    int x = 0;
    int z = 0;
    while (y > 0) {
        x++;
        z = z + y % 10;
        y = y / 10;
    }
    System.out.println(x + " " + z);
}
```

For each call below, indicate what output is produced.

Method Call	Output Produced
mystery(8);	_____
mystery(32);	_____
mystery(184);	_____
mystery(8239);	_____

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) {
    if (x <= 0) {
        x = 42;
    }
    int y = 0;
    // Point A
    while (x != 1) {
        // Point B
        if (x % 2 == 0) {
            y++;
            x = x / 2;
            // Point C
        } else {
            x = 3 * x + 1;
            // Point D
        }
    }
    // Point E
    return y;
}
```

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

	x == 1	x % 2 == 1	y == 0
Point A			
Point B			
Point C			
Point D			
Point E			

6. Programming, 15 points. Write a static method called generate that takes a console Scanner as a parameter and that asks a user to pick a number between -5 and 5 inclusive, showing a sequence of random numbers in that range until the user's number comes up and returning how many numbers it generated. The method should construct and use a Random object to generate the numbers. For example, if you execute the following code:

```
Scanner console = new Scanner(System.in);
int result = generate(console);
```

you would see an interaction like this (user input bold and underlined):

```
number between -5 and 5? 0
numbers are: 5, 5, -2, -1, -2, 3, -4, -5, 2, 0
came up after 10 tries
```

Notice that the user is prompted for a number and enters 0. The method then generates a random sequence of numbers in that range until the user's number comes up, showing the sequence of numbers. Then it reports and returns how many numbers it generated. The variable result would be set to 10. It is possible that the user's number will be generated immediately, as in (user input bold and underlined):

```
number between -5 and 5? -5
numbers are: -5
came up after 1 tries
```

Here the user enters -5, which is the first value generated, so the method reports that and returns 1. You are to exactly reproduce the format of these logs.

7. Programming, 15 points. Write a static method called digitRange that takes an integer n as a parameter and that returns the greatest difference between two digits of n. In particular, the method should report the largest difference (x - y) where x and y are digits of n. For example, the call digitRange(68437) should return 5 because the largest difference that can be formed using digits from the number is (8 - 3). You may assume that the number passed to the method is greater than or equal to 0. If the method is passed a 1-digit number, it should return 0. Below are more examples of calls and the values that should be returned.

Method Call	Value Returned	Method Call	Value Returned
digitRange(0)	0	digitRange(888)	0
digitRange(5)	0	digitRange(1234)	3
digitRange(26)	4	digitRange(24680)	8
digitRange(42)	2	digitRange(857492)	7
digitRange(725)	5	digitRange(3876254)	6

You are not allowed to use a String to solve this problem.

8. Programming, 9 points. Write a static method called printStripped that takes a string as a parameter and that prints a complete line of output with any comments stripped from the string. Comments are defined to be characters enclosed in the characters "<" and ">". More precisely, text is "normal" until you encounter a "<" character. From that point on the text is considered a comment until you encounter a ">" character, at which point you return to normal text. This definition allows for "<" inside a comment and ">" outside a comment. You may assume that there are no unclosed comments in the string.

   For example, the following sequence of calls:

```
printStripped("this is plain text");
printStripped("this has a normal comment <right here> to be removed");
printStripped("this has multiple less-than in a comment <<<<<see?>");
printStripped("this > has <comment>greater-than outside a comment >>");
printStripped("<hi>this has <foo> multiple <bar> comments<baz><>.");
```

   should produce the following output:

```
this is plain text
this has a normal comment  to be removed
this has multiple less-than in a comment
this > has greater-than outside a comment >>
this has  multiple  comments.
```

   In solving this problem you may use only methods listed on the cheat sheet.