CSE142 Sample Midterm
Autumn 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
`3 * 4 + 5 * 6`	_____
`23 % 5 - 17 % (16 % 10)`	_____
`"1" + 2 + 3 * 4 + (5 + 6)`	_____
`1.5 * 2 + 20 / 3 / 4.0 + 6 / 4`	_____
`345 / 10 / 3 + 10 / (5 / 2.0)`	_____

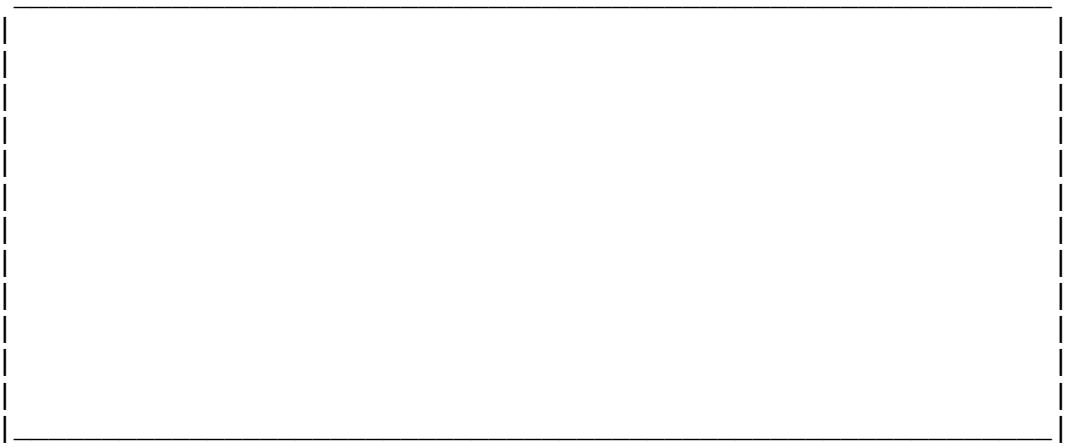
2. Parameter Mystery, 12 points. Consider the following program.

```
public class ParameterMystery {
    public static void main(String[] args) {
        String literal = "8";
        String brace = "semi";
        String paren = brace;
        String semi = "brace";
        String java = "42";

        param(java, brace, semi);
        param(literal, paren, java);
        param(brace, semi, "literal");
        param("cse", literal + 4, "1");
    }

    public static void param(String semi, String java, String brace) {
        System.out.println(java + " missing a " + brace + " and a " + semi);
    }
}
```

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) {
        a = a + 10;
    } else if (a > b) {
        b++;
    }
    if (b < a) {
        a--;
    } else {
        b = b + 10;
    }
    System.out.println(a + " " + b);
}
```

For each call below, indicate what output is produced.

Method Call	Output Produced
ifElseMystery(2, 7);	_____
ifElseMystery(6, 6);	_____
ifElseMystery(4, -1);	_____
ifElseMystery(11, 10);	_____
ifElseMystery(-10, 7);	_____
ifElseMystery(100, 5);	_____

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

```
public static void mystery(int n) {
    int x = 1;
    int y = 1;
    while (n > y) {
        x++;
        y = 10 * y + x;
    }
    System.out.println(x + " " + y);
}
```

For each call below, indicate what output is produced:

Method	Output Produced
mystery(0);	_____
mystery(7);	_____
mystery(32);	_____
mystery(256);	_____

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.

```
// Reads many integers from Scanner; returns the second smallest
// integer read.
public static int secondSmallest(Scanner console) {
    int num = console.nextInt();
    int first = num;
    int second = num;
    // Point A
    while (num >= 0) {
        // Point B
        if (num < first) {
            second = first;
            first = num;
            // Point C
        } else if (num < second) {
            second = num;
        }
        // Point D
        num = console.nextInt();
    }
    // Point E
    return second;
}
```

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

	num < 0	first < second	num >= second
Point A			
Point B			
Point C			
Point D			
Point E			

6. Programming, 15 points. Write a static method called spinWheel that takes a Random object and an integer n as parameters and that simulates the spinning of a wheel until the number 20 comes up n times in a row. On the wheel are the numbers 20, 30, 40, 50, and 60 and each number should be equally likely to come up when the wheel is spun. Your method should report the individual spins as well as indicating the number of spins it took to get n occurrences of 20 in a row. For example, below are two sample calls:

```
Random r = new Random();
spinWheel(r, 2);
spinWheel(r, 3);
```

The first call should produce two lines of output like this:

```
spins: 40, 40, 50, 20, 50, 50, 40, 20, 30, 40, 50, 20, 20
2 in a row after 13 spins
```

The second call should produce two lines of output like this:

```
spins: 50, 50, 50, 20, 40, 20, 40, 20, 20, 20
3 in a row after 10 spins
```

Notice that the spin values are separated by commas and that the method stops when it has seen n occurrences of the value 20 in a row. You are to exactly reproduce the format of these logs. You may assume that the value n passed to your method is greater than or equal to 1.

7. Programming, 15 points. Write a static method called balanceCheckbook that takes a console Scanner as a parameter and that prompts a user for information about transactions for a bank account, reporting the new balance after each transaction and the minimum balance at the end and returning whether or not the account was ever overdrawn (true if the balance ever becomes negative, false if it never did). The user is prompted for an initial balance, the number of transactions to process, and the individual transaction amounts. Deposits to the account are entered as positive numbers and checks and ATM withdrawals are entered as negative numbers. A new balance is reported after each transaction. For example, the method would be called as follows:

```
Scanner console = new Scanner(System.in);
boolean wasOverdrawn = balanceCheckbook(console);
```

Below are two sample logs of execution that might be produced, where user input is **bolded** and underlined:

```
initial balance? 48.50
how many transactions? 4
1/4 amount? -20.00
new balance = $28.5
2/4 amount? -5.75
new balance = $22.75
3/4 amount? 138.20
new balance = $160.95
4/4 amount? -20.00
new balance = $140.95
minimum balance = $22.75
```

```
initial balance? 39.75
how many transactions? 5
1/5 amount? -18.50
new balance = $21.25
2/5 amount? -7.20
new balance = $14.05
3/5 amount? -23.10
new balance = $-9.05
4/5 amount? 50.00
new balance = $40.95
5/5 amount? -8.45
new balance = $32.5
minimum balance = $-9.05
```

In the log to the left, the user enters 4 different transactions and the minimum balance is not negative, so the method would return false to indicate that the account was never overdrawn. In the log to the right, the user enters 5 transactions and the minimum balance is negative, so the method would return true to indicate that the account was overdrawn at some point. You are to exactly reproduce the format of these logs. You may assume that the number of transactions entered by the user is at least 1.

8. Programming, 9 points. Write a static method called hashTag that takes a String as a parameter and returns the String converted to a hashtag. A hashtag is a String made up of a pound sign (#) followed by a phrase, where each word is capitalized (with the first letter uppercase, and the other letters lowercase). Words in the original string are separated by one or more spaces. For example, the call hashTag("I love computer science") would return "#ILoveComputerScience"

Below are more sample calls:

Method Calls	Value Returned
hashTag("I love computer science")	"#ILoveComputerScience"
hashTag("to be or not to be")	"#ToBeOrNotToBe"
hashTag("saY YES")	"#SayYes"
hashTag(" edGAr allan pOE ")	"#EdgarAllanPoe"
hashTag(" sPooOOOooOoOky")	"#Spooooooooooky"
hashtag(" fuNNY #@*^!& sYMbols ")	"#Funny#@*^!&Symbols"
hashTag("x")	"#X"
hashTag(" ")	"#"
hashTag("")	"#"

Notice that words can contain other punctuation. Any non-empty sequence of non-space characters can be a word. Also notice that there may be spaces at the beginning or end of the String. You may not construct any Scanners or tokenizers to solve this problem. You may assume that the String has no other whitespace characters such as tabs or newline characters.

In addition to the String methods on your cheat sheet, you may use the methods Character.toUpperCase() and Character.toLowerCase(), which take a char as a parameter and returns the char in uppercase and lowercase, respectively. For example, Character.toUpperCase('a') returns 'A'. If the character is already in uppercase or is not a letter, it returns the character unchanged (e.g., Character.toUpperCase('A') returns 'A' and Character.toUpperCase('@') returns '@').