CSE142 Section Handout #5.5 Questions

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).

Expression	Value
`13 + 2 * 5 / 3`	_____
`2.5 * 2 * 5 / 10 + 1.5`	_____
`85 % 10 + 4 % 10 - 17 % 3`	_____
`2 + 3 + "." + (3 + 4) + 2 * 3`	_____
`482 / 10 / 5 / 2.0 * 2 + 14 / 5`	_____

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

```
public class Mystery {
    public static void main(String[] args) {
        String john = "skip";
        String mary = "george";
        String george = "mary";
        String fun = "drive";

        statement(mary, john, fun);
        statement(fun, "george", "work");
        statement(george, mary, john);
        statement(george, "john", "dance");
    }

    public static void statement(String mary, String john, String fun) {
        System.out.println(john + " likes to " + fun + " with " + mary);
    }
}
```

List below the output produced by this program.

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

```
public static void ifElseMystery(int a, int b) {
    if (a == b) {
        b--;
    } else if (a < b) {
        a++;
    } else {
        b = b + 5;
    }
    if (a == b) {
        a = a + 2;
    }
    System.out.println(a + " " + b);
}
```

For each call below, indicate what output is produced.

Method Call	Output Produced
ifElseMystery(14, 14);	_____
ifElseMystery(4, 5);	_____
ifElseMystery(10, 5);	_____
ifElseMystery(2, 8);	_____

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(Scanner console) {
    int prev = 0;
    int count = 0;
    int next = console.nextInt();
    // Point A
    while (next != 0) {
        // Point B
        if (next == prev) {
            // Point C
            count++;
        }
        prev = next;
        next = console.nextInt();
        // Point D
    }
    // Point E
    return count;
}
```

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

	next == 0	prev == 0	next == prev
Point A			
Point B			
Point C			
Point D			
Point E			

6. Programming, 15 points. Write a method called numUnique that takes three integers as parameters and that returns the number of unique integers among the three. For example, if the following call is made:

```
numUnique(18, 3, 4)
```

the method should return 3 because the parameters represent 3 different numbers (the values 18, 3 and 4). By contrast, the following call:

```
numUnique(6, 6, 6)
```

would return 1 because there is only 1 unique number among the three parameters (the value 6). The values passed to your method might appear in any order whatsoever, so you cannot assume that duplicate values would appear together. For example, if the following call is made:

```
numUnique(7, 31, 7)
```

the method should return 2 because there are 2 unique numbers among the parameters (the values 7 and 31).

7. Programming, 15 points. Write a method called printNumbers that takes a Random object as a parameter and that prints a list of randomly generated numbers ranging from 1 to 50 inclusive (each number being equally likely). The list should be surrounded by square brackets and the numbers should be separated by commas. The method should randomly generate numbers until it generates one that ends in 5. A typical call would look like this:

```
Random r = new Random();
printNumbers(r);
```

The method might fairly quickly generate a number ending in 5:

```
[43, 34, 27, 2, 2, 25]
```

Or it might take a while to get to a number ending in 5:

```
[23, 8, 13, 1, 37, 37, 9, 34, 23, 34, 4, 9, 16, 44, 49, 43, 49, 3, 45]
```

It is also possible that it will immediately generate a number ending in 5:

```
[35]
```

You must exactly reproduce the format of these examples.

8. Programming, 9 points. Write a method called numWords that takes a String as a parameter and that returns the number of words in the String. By definition, words are separated by one or more spaces. The table below shows several sample calls and the value that should be returned.

Method Call	Value Returned
numWords("how many words here?")	4
numWords("to be or not to be, that is the question")	10
numWords(" how about merry-go-round ")	3
numWords(" !&$%--$$!!*() foo_bar_baz ")	2
numWords("x")	1
numWords(" ")	0
numWords("")	0

Notice that words can contain punctuation marks. Any non-empty sequence of non-space characters can be a word. Also notice that there might be spaces at the beginning or end of the String.

You may not construct any other objects to solve this problem (e.g., you can't use a Scanner or tokenizer). You may assume that the String has no other whitespace characters such as tabs or newline characters. Your method can pay attention just to spaces to decide how many words there are.

CSE142 Section Handout #5.5 Solutions

1.

Expression	Value
`13 + 2 * 5 / 3`	16
`2.5 * 2 * 5 / 10 + 1.5`	4.0
`85 % 10 + 4 % 10 - 17 % 3`	7
`2 + 3 + "." + (3 + 4) + 2 * 3`	"5.76"
`482 / 10 / 5 / 2.0 * 2 + 14 / 5`	11.0

2. Parameter Mystery. The program produces the following output.

```
skip likes to drive with george
george likes to work with drive
george likes to skip with mary
john likes to dance with mary
```

3.

Method Call	Output Produced
`ifElseMystery(14, 14);`	14 13
`ifElseMystery(4, 5);`	7 5
`ifElseMystery(10, 5);`	12 10
`ifElseMystery(2, 8);`	3 8

4.

Method Call	Output Produced
`mystery(8);`	1 8
`mystery(32);`	2 5
`mystery(184);`	3 13
`mystery(8239);`	4 22

5.

	next == 0	prev == 0	next == prev
Point A	sometimes	always	sometimes
Point B	never	sometimes	sometimes
Point C	never	never	always
Point D	sometimes	never	sometimes
Point E	always	sometimes	sometimes

6. Two possible solutions appear below.

```
public static int numUnique(int x, int y, int z) {
    if (x == y && y == z) {
        return 1;
    } else if (x != y && x != z && y != z) {
        return 3;
    } else {
        return 2;
    }
}
```

```
public static int numUnique(int x, int y, int z) {
    int count = 0;
    if (x != y) {
        count++;
    }
    if (y != z) {
        count++;
    }
    if (x != z) {
        count++;
    }
    if (count == 0) {
        count = 1;
    }
    return count;
}
```

7. One possible solution appears below.

```
public static void printNumbers(Random r) {
    int n = r.nextInt(50) + 1;
    System.out.print("[" + n);
    while (n % 10 != 5) {
        n = r.nextInt(50) + 1;
        System.out.print(", " + n);
    }
    System.out.println("]");
}
```

8. Two possible solutions appear below.

```
public static int numWords(String s) {
    int count = 0;
    boolean inWord = false;
    for (int i = 0; i < s.length(); i++) {
        if (s.charAt(i) == ' ') {
            inWord = false;
        } else if (!inWord) {
            count++;
            inWord = true;
        }
    }
    return count;
}
```

```
public static int numWords(String s) {
    int count = 0;
    for (int i = 1; i < s.length(); i++) {
        if (s.charAt(i - 1) == ' ' && s.charAt(i) != ' ') {
            count++;
        }
    }
    if (s.length() > 0 && s.charAt(0) != ' ') {
        count++;
    }
    return count;
}
```