CSE 142 Section Handout #5
Questions

While Loop Mystery

1. Self-Check 5.3, p378. Consider the following method. For each call below, indicate what output is produced.

```
public static void mystery1(int x) {
    int y = 1;
    int z = 0;
    while (2 * y <= x) {
        y = y * 2;
        z++;
    }
    System.out.println(y + " " + z);
}
```

Call	Output
`mystery1(1);`	_____
`mystery1(6);`	_____
`mystery1(19);`	_____
`mystery1(39);`	_____
`mystery1(74);`	_____

2. Consider the following method. For each call below, indicate what value is returned.

```
public static int mystery2(int x) {
    int a = 1;
    int c = 0;
    while (x > 0) {
        a = x % 2;
        if (a == 1) {
            c++;
        }
        x = x / 2;
    }
    return c;
}
```

Call	Value Returned
`mystery2(2);`	_____
`mystery2(-1);`	_____
`mystery2(7);`	_____
`mystery2(18);`	_____
`mystery2(43);`	_____

3. Self-Check 5.20, p382. Consider the following method. For each call below, indicate what value is returned.

```
public static int mystery3(int x, int y) {
    while (x != 0 && y != 0) {
        if (x < y) {
            y = y - x;
        } else {
            x = x - y;
        }
    }
    return x + y;
}
```

Call	Value Returned
`mystery3(3, 3)`	_____
`mystery3(5, 3)`	_____
`mystery3(2, 6)`	_____
`mystery3(12, 18)`	_____
`mystery3(30, 75)`	_____

While Loop Programming

4. Exercise 5.1, p387. Write a static method `showTwos` that shows the factors of 2 in an integer. For example:

Call	Output
`showTwos(7);`	`7 = 7`
`showTwos(18);`	`18 = 2 * 9`
`showTwos(68);`	`68 = 2 * 2 * 17`
`showTwos(120);`	`120 = 2 * 2 * 2 * 15`

The idea is to express the number as a product of factors of 2 and an odd number. The number 120 has 3 factors of 2 multiplied by the odd number 15. For odd numbers (e.g. 7), there are no factors of 2, so you just show the number itself. Assume that your method is passed a number greater than 0.

CSE 142 Section Handout #5
Questions (continued)

While Loop Programming

5. Write a method `showHailstone` that takes an integer parameter *n* and that displays the hailstone sequence starting at n and ending with 1. In a hailstone sequence, each value x is followed either by:

$3x + 1$ *if x is odd*
$x/2$ *if x is even*

Below are a series of calls and the output produced:

Call	**Output**
`showHailstone(3);`	`sequence for 3: 3, 10, 5, 16, 8, 4, 2, 1`
`showHailstone(10);`	`sequence for 10: 10, 5, 16, 8, 4, 2, 1`
`showHailstone(1);`	`sequence for 1: 1`

It is believed that for any positive integer n, the sequence always reaches 1, although nobody has yet proven that this is true. Assume that your method is passed a number greater than 0.

Boolean Logic

6. Write a method named `sign` that accepts two integers as parameters, and that returns a `String` indicating the sign of the result of multiplying the integers together. Your method should either return `"Positive"`, `"Negative"`, or `"Zero"`. Do not perform any arithmetic operations on the integers (`+`, `-`, `*`, `/`). Example calls and return values are listed below.

Call	**Return Value**
`sign(4, 0);`	`"Zero"`
`sign(2, -3);`	`"Negative"`

Random Numbers

7. Write a method named `rollSix` that simulates the repeated rolling of one six-sided die until a six is rolled. You should use a `Random` object to give an equal chance of rolling a one through six. Each time the die is rolled, you should display the number seen. When you roll a six, you should print the number of trials taken. An example output of a call to `rollSix` is shown below:

```
Rolled: 3
Rolled: 1
Rolled: 6
You got a six in 3 turns!
```

8. **Exercise 5.8, p388.** Write a method named `randomWalk` that performs a random one-dimensional walk, reporting each position reached and the maximum position reached during the walk. The random walk should begin at position 0. On each step, you should either increase or decrease the position by 1 (with equal probability). The walk stops when 3 or -3 is hit. The output should look like this:

```
position = 0
position = 1
position = 0
position = -1
position = -2
position = -1
position = -2
position = -3
max position = 1
```

CSE 142 Section Handout #5
Style Sheet

This program prompts the user for a value and then generates random numbers between 0 and 99 until it finds a number divisible by the given number, as in:

```
number to use? 10
Let's find a number divisible by 10
83, 88, 32, 8, 33, 21, 95, 76, 89, 78, 2, 99, 1, 20
found one after 14 tries
```

Consider the following implementation of the program:

```
import java.util.*;

public class Sect5 {
   public static void main(String[] args) {
      Scanner console = new Scanner(System.in);
      System.out.print("number to use? ");
      int a = console.nextInt();
      gN(console, a);
   }

   // generates random numbers, printing numbers separated by commas
   public static void gN(Scanner console, int a) {
      System.out.println("Let's find a number divisible by " + a);
      int number = 1;
      int count = 0;
      while (number % a != 0) {
         Random r = new Random();
         number = r.nextInt(100);
         System.out.print(number);
         if (number % a != 0) {
            count++;
            System.out.print(", ");
         } else if (number % a == 0) {
            count++;
            System.out.println();
            System.out.println("found one after " + count + " tries");
         }
      }
   }
}
```

While this program would receive full external correctness by producing the desired output, it would not receive full internal correctness. List all style issues you can find.