CSE142 Sample Midterm
Spring 2018

Name ____________________________________ UW NetId (e.g. whitab) ______________

Section (e.g., AA) _____________ TA ____

This exam is divided into nine questions with the following points:

#	Problem Area	Points	Score
1	Expressions	10	_____
2	Parameter Mystery	12	_____
3	If/Else Simulation	12	_____
4	While Loop Simulation	12	_____
5	Assertions	15	_____
6	Programming	15	_____
7	Programming	15	_____
8	Programming	9	_____
9	Prognostication (bonus)	1	_____
	Total	100	_____

This is a closed-book/closed-note exam. Space is provided for your answers. You can also request scratch paper from a TA. You are not allowed to access any of your own papers during the exam. The exam is not graded on style and you do not need to include comments, although you are limited to the constructs included in chapters 1 through 5 of the textbook. You are not required to include any import statements; you may assume standard classes are imported.

You are allowed to abbreviate "Always", "Never," and "Sometimes" as "A", "N", and "S" for the assertions question, but you should otherwise NOT use any abbreviations on the exam.

You are NOT to use any electronic devices while taking the test, including calculators and smart watches. Anyone caught using an electronic device will receive a 10-point penalty. Do not begin work on this exam until instructed to do so. Any student who starts early or who continues to work after time is called will receive a 10-point penalty.

If you finish the exam early, please hand your exam to a TA and exit quietly.

Initial here to indicate you have read and agree to these rules. If you do not initial, your exam may not be accepted for credit:

```
 _______________________
|                       |
|                       |
|                       |
|_______________________|
```

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
12 + 3 / 5 + 3 % 2	________________
7 + 1 + "4 + 2" + 1 + 7	________________
15 / 4 / 3.0 - 18 / 5 + (15 / 10.0)	________________
!(7 * 2 != 42 && !(5 / 2 == 2))	________________
6 % 4 + 4 % 6 + 6 % 6	________________

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

```
public class ParameterMystery {
   public static void main(String[] args) {
      String be = "to";
      String not = "or";
      String or = "ophelia";
      String hamlet = "be";
      String to = not;

      hamlet(be, "or", or);
      hamlet("not", hamlet, to);
      hamlet(to, not, be);
      hamlet += "?";
      hamlet(be, hamlet, "not");
   }

   public static void hamlet(String to, String be, String not) {
      System.out.println(to + " be or " + not + " to " + be);
   }
}
```

List below the output produced by this program.

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

```
public static void ifElseMystery(int one, int two) {
    if (one % two == 0 || one % two == 1) {
        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(40, 5);      _______________

ifElseMystery(64, 8);      _______________

ifElseMystery(12, 12);     _______________

ifElseMystery(13, 3);      _______________

ifElseMystery(122, 6);     _______________
```

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

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

For each call below, indicate what output is produced.

```
Method Call              Output Produced
---------------------------------------
mystery(1);              _______________

mystery(4);              _______________

mystery(6);              _______________
```

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 nonsense(int x) {
    Scanner console = new Scanner(System.in);
    System.out.print("Enter a number GREATER than " + x + ": ");
    int y = console.nextInt();
    y = y * 2;

    // Point A
    while (y > x) {
        // Point B
        if (x % 2 == 1) {
            x++;
            y--;
        } else if (y % 2 == 0) {
            // Point C
            y /= 2;
        } else {
            y++;
            x = x - 2;
            // Point D
        }

    }

    // Point E
    return x;
}
```

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

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

6. Programming, 15 points. It is said that when people talk to dogs, the dog only hears its name, interpreting any other word as nonsense. Write a static method called dogHears that converts human speech to what a dog hears. The method accepts three parameters: the dog's name (as a String), a number of words (as an int), and a Scanner (for user input). Your method should use the Scanner to read in the given number of words and print out what a dog with the given name hears when that word is said. If the word exactly matches the dog's name (including casing), the dog hears its name. Otherwise, the dog hears the word "blah". After the given number of words have been entered and translated, your method should return the number of times the dog's name was heard.

For example, if the following calls were made:

```
Scanner console = new Scanner(System.in);
int numFido = dogHears("Fido", 10, console);
```

we would expect interaction like the following (user input bold and underlined):

```
word? Fido
dog hears: "Fido"
word? is
dog hears: "blah"
word? the
dog hears: "blah"
word? best
dog hears: "blah"
word? dog
dog hears: "blah"
word? Fido
dog hears: "Fido"
word? is
dog hears: "blah"
word? better
dog hears: "blah"
word? than
dog hears: "blah"
word? Spot
dog hears: "blah"
```

In this case, the method would return the value 2. Your method must exactly reproduce the format of this log.

7. Programming, 15 points. Write a static method called walkHome that simulates a confused bug trying to find its way home. Your method should take two parameters: an integer that represents the bug's starting position relative to its home and a Random object. The bug should repeatedly move a random integer number of steps between -2 and 2 (inclusive) with all values equally likely. Positive steps will move the bug closer to its home and negative steps will move it farther away. After each move, you should print how many steps the bug moved and a representation of its position as seen in the format below. In the output, the asterisk (*) represents the bug and the carat (|^|) represents the bug's home. The bug should continue moving random amounts until it reaches home, at which point the total number of steps taken should be printed.

For example, if the following calls were made:

```
Random rand = new Random();
walkHome(2, rand);
```

we would expect output that looks like the following:

```
starting at 2
*--|^|
moving -1 step(s)
*---|^|
moving -2 step(s)
*-----|^|
moving 2 step(s)
*---|^|
moving -2 step(s)
*-----|^|
moving 0 step(s)
*-----|^|
moving 1 step(s)
*----|^|
moving 1 step(s)
*---|^|
moving 1 step(s)
*--|^|
moving 2 step(s)
*|^|
made it home in 12 step(s)
```

If the following subsequent calls were made:

```
rand = new Random();
walkHome(0, rand);
```

we would expect the following output:

```
starting at 0
*|^|
made it home in 0 step(s)
```

The bug should never move past its home. If you randomly select a number of steps that would move the bug past its home, you should limit the number of steps to the amount it needs to get home. You may assume that the integer argument passed to your method is always greater than or equal to zero. Your method must exactly reproduce the format of this log, though the actual output may be different due to randomness.

8. Programming, 9 points. Write a static method called digitsInARow that takes an integer n as a parameter and that returns the highest number of digits that appear in a row in the base-10 representation of n. For many numbers the answer will be 1 because they don't have adjacent digits that match. But for a number like 3555585, the answer is 4 because there are four occurrences of the digit 5 that appear in a row. Below are sample calls to the method.

Method Call	Value Returned
digitsInARow(0)	1
digitsInARow(8823)	2
digitsInARow(18)	1
digitsInARow(777)	3
digitsInARow(394)	1
digitsInARow(82888)	3
digitsInARow(99)	2
digitsInARow(7111171)	4
digitsInARow(8229)	2
digitsInARow(233333888)	5
digitsInARow(100)	2

You may assume that the integer passed as a parameter to your method is greater than or equal to 0. You may not use Strings to solve this problem.

9. Prognostication, 1 point (bonus). Predict your final score on this exam.
   Note that accuracy is not required for credit.