Name : _____

EGR222 Practice Exam Questions

1. Scope: Lecture notes, videos, HWs, PIs, textbook Ch 1- Ch 7 (including 3G graphics)

2. Closed book + instructor provided cheat sheet

3. 90 minutes written test (pencil and paper)

1. Trace the evaluation of the following expressions, and give their resulting values.

```
12 / 3 + 5 + 3 * -2
1 + 1 + "(1 + 1)" + 1 + 1
13 / 2 - 38 / 5 / 2.0 + (15 / 10.0)
11 < 3 + 4 || !(5 / 2 == 2)
20 % 6 + 6 % 20 + 6 % 6
```

2. Write the output of each of the method calls in the following program, as it would appear on the console

```
public class ParameterMystery {
    public static void main(String[] args) {
        int a = 5;
        int b = 1;
        int c = 3;
        int three = a;
        int one = b + 1;

        axiom(a, b, c);
        axiom(c, three, 10);
        axiom(b + c, one + three, a + one);
        a++;
        b = 0;
        axiom(three, 2, one);
    }

    public static void axiom(int c, int b, int a) {
        System.out.println(a + " + " + c + " = " + b);
    }
}
```

```
axiom(a, b, c);
axiom(c, three, 10);
axiom(b + c, one + three, a + one);
axiom(three, 2, one);
```

3. For each call of the method below, write the output that is produced:

```
public static void ifElseMystery(int x, int y) {
    if (x == y) {
        x = x + 11;
    } else if (x > 2 * y) {
        x = 0;
    }
    if (x == 0 || y > x) {
        x = x + 2;
        y = y + 2;
    }

    System.out.println(x + " " + y);
}
```

```
ifElseMystery(5, 5);
ifElseMystery(18, 4);
ifElseMystery(3, 6);
```

4. For each call of the method below, write the output that is produced:

```
public static void whileMystery(int x, int y) {
    while (x > 0 && y > 0) {
        x = x - y;
        y--;
        System.out.print(x + " ");
    }

    System.out.println(y);
}
```

```
whileMystery(7, 5);
whileMystery(20, 4);
whileMystery(40, 10);
```

5. For each array below, indicate what the array's contents would be after the method `mystery` were called and passed that array as its parameter.

```
public static void arrayMystery5(int[] nums) {
    for (int i = 0; i < nums.length - 1; i++) {
        if (nums[i] > nums[i + 1]) {
            nums[i + 1]++;
        }
    }
}
```

{8}	
{14, 7}	
{7, 1, 3, 2, 0, 4}	
{10, 8, 9, 5, 5}	
{12, 11, 10, 10, 8, 7}	

6. What are the values of the elements in the array numbers after the following code is executed?

```
int[] numbers = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
for (int i = 0; i < 9; i++) {
    numbers[i] = numbers[i + 1];
}
```

numbers[0]	
numbers[1]	
numbers[2]	
numbers[3]	
numbers[4]	
numbers[5]	
numbers[6]	
numbers[7]	
numbers[8]	
numbers[9]	

7. Given the following file contents, what will be the output from each of the following code fragments?

```
the quick brown
   fox   jumps

       over
the lazy    dog
```

```
// a.
Scanner input = new Scanner(new File("brownfox.txt"));
while (input.hasNextLine()) {
    String line = input.nextLine();
    System.out.println(line);
}
// b.
Scanner input = new Scanner(new File("brownfox.txt"));
while (input.hasNext()) {
    String token = input.next();
    System.out.println(token);
}
```

a.

b.

8. Fill in below blank where randomNumber stores a random odd number ranging from 101 to 199 inclusive.

int randomNumber = _____ ;

9. Explain why main is declared with “public static void” modifiers.

1) public
2) static
3) void

10. Which modifier(s) should be used for declaring class constant? Explain the reason for each modifier if there is more than one.

11. Why chaining method is bad?

12. 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. You are NOT allowed to use a `String` or array to solve this problem. You may assume that the value passed to the method is greater than or equal to 0.

13. In this problem, we'll address the following question: Can a cash register containing a given amount of pennies (1-cent coins) and a given amount of nickels (5-cent coins) give a customer a given exact amount of cents of change? For example, if there are 3 pennies and 5 nickels in the cash register, is it possible to give exactly 19 cents of change? (No.) If there are 2 pennies and 7 nickels in the register, is it possible to give exactly 26 cents of change? (Yes.)

Write a method named `canMakeChange` that accepts three integer parameters representing the number of pennies in the cash register, the number of nickels in the cash register, and the desired amount of change to make. The method should return `true` if the coins in the register could be used to produce this exact amount of change, and `false` if not. For example, if the register contains 3 pennies and 4 nickels, it is able to exactly produce 17 cents of change (using 2 of the pennies and 3 of the nickels), so the call of `canMakeChange(3, 4, 17)` should return `true`. If the register contains 1 penny and 10 nickels, it is not able to exactly produce 8 cents of change (in this case, there are not enough pennies to reach exactly 8), so the call of `canMakeChange(1, 10, 8)` should return `false`.

The following are several sample calls to your method and the values they should return. You may assume that no negative parameter values are passed, but otherwise your method should work with any values passed.

Call	Value Returned
`canMakeChange(3, 4, 12)  // 3 pennies, 4 nickels, 12c change?`	`true`
`canMakeChange(1, 5, 26)  // 1 penny, 5 nickels, 26c change?`	`true`
`canMakeChange(24, 2, 31)  // 24 pennies, 2 nickels, 31c change?`	`true`
`canMakeChange(87, 19, 134)  // 87 pennies, 19 nickels, 134c change?`	`true`
`canMakeChange(0, 0, 0)  // 0 pennies, 0 nickels, 0c change?`	`true`
`canMakeChange(1, 1, 9)  // 1 penny, 1 nickel, 9c change?`	`false`
`canMakeChange(2, 7, 8)  // 2 pennies, 7 nickels, 8c change?`	`false`
`canMakeChange(4, 3, 39)  // 4 pennies, 3 nickels, 39c change?`	`false`
`canMakeChange(3, 80, 14)  // 3 pennies, 80 nickels, 14c change?`	`false`

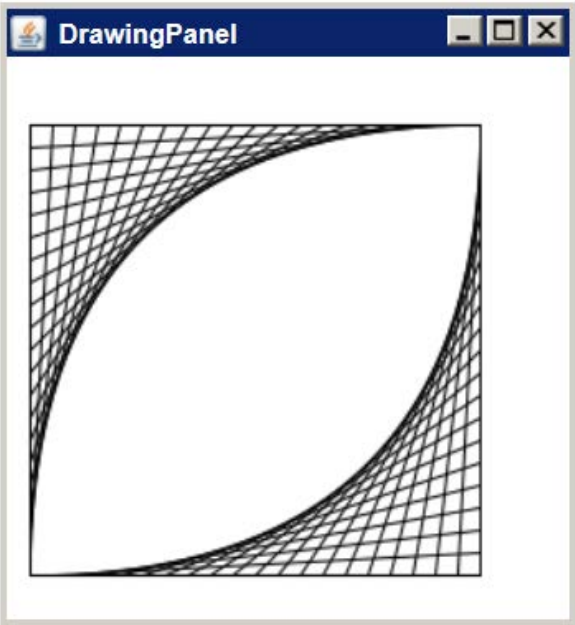
14. Write a method named `randomRects` that calculates and displays the area of randomly-generated rectangles. The width and height of each rectangle should be a randomly generated integer between 1 and 10 inclusive. Your method should keep generating random rectangles until an increasing sequence of four areas is printed. In other words, if the last four rectangles generated have areas of *a1*, *a2*, *a3* and *a4* such that *a1* < *a2* < *a3* < *a4*, the method should print the final message and stop. So your method will generate at least 4 total rectangles but possibly many more, stopping only when it sees 4 in a row with areas in increasing order.

The following calls demonstrate your method's behavior. Your method should match this output format exactly:

Call:	`randomRects();`	`randomRects();`

Output:	`w: 5, h: 6, area: 30`
	`w: 10, h: 5, area: 50`
	`w: 2, h: 8, area: 16`
	`w: 4, h: 4, area: 16`
	`w: 2, h: 9, area: 18`
	`w: 8, h: 3, area: 24`
	`w: 7, h: 2, area: 14`
	`w: 3, h: 10, area: 30`
	`w: 7, h: 9, area: 63`
	`w: 9, h: 8, area: 72`
	`Four rectangles of increasing area.`
		`w: 5, h: 2, area: 10`
		`w: 6, h: 5, area: 30`
		`w: 7, h: 6, area: 42`
		`w: 8, h: 10, area: 80`
		`Four rectangles of increasing area.`

15. Using the `DrawingPanel` class, write a Java class named `Football` that produces the following figure:



Though the figure looks to contain curves, it is made entirely of straight lines. The window is 250 x 250 pixels in size, and there is an outer rectangle from (10, 30) to (210, 230). A set of black lines are drawn around the edges every 10 pixels. For example, along the top-left, there is a line from (10, 220) to (20, 30), a line from (10, 210) to (30, 30), a line from (10, 200) to (40, 30), ... and so on. Along the bottom-right, there is a line from (20, 230) to (210, 220), a line from (30, 230) to (210, 210), and so on.

```
public class Football {
   public static void main(String[] args) {









   }
}
```