Midterm #2 Study Guide, Fall 2015

This midterm will cover:

a. Chapters 1-5, 7
b. All the materials covered in class
c. All the problems solved in class
d. Practice-it
e. projects

These are some sample questions for you to practice. The difficulty level of your exam will be the same as the following questions. Make sure to go over the questions before the next class meeting, come ready to the next class to ask your questions.

1. Given the following variable declarations:

```
int x = 4;
int y = -3;
int z = 4;
```

What are the results of the following relational expressions?

`x == 4 % 3 * 2`	
`x == y / 3 % 3 + 5`	
`x == z`	
`y == z`	
`x + y > 0 + y && x == z + y -y`	
`!(x - z != 0)`	
`y * y <= z`	
`y / y == 1 || x + y == 1 && !(x * y <0)`	
`!(x * (y + 2) > y - (y + z) * 2)`	

2. What output is produced by the following program?

```
public class CharMystery {
    public static void printRange(char startLetter, char endLetter) {
        for (char letter = startLetter; letter <= endLetter; letter++) {
            System.out.print(letter);
        }
        System.out.println();
    }

    public static void main(String[] args) {
        printRange('e', 'g');
        printRange('n', 's');
        printRange('z', 'a');
        printRange('q', 'r');
```

```
    }
}
```

```
printRange('e', 'g');
printRange('n', 's');
printRange('z', 'a');
printRange('q', 'r');
```

3. Write a method called median that accepts 3 integer parameters and returns the median of the three values. For example the medianOf3(3,4,1) should return 3, medianOf3(7,2,45) should return 7. No array can be used.

```
4. public static void ifElseMystery1(int x, int y) {
    int z = 8;
    if (z <= x) {
        z = x++ + 1;
    } else {
        z = --z + 9;
    }

    if (z <= y) {
        y++;
    }
    else{
       y--;
    }

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

For each call below, indicate what output is produced.

```
ifElseMystery1(3, 20);
ifElseMystery1(4, 5);
ifElseMystery1(5, 5);
ifElseMystery1(6, 10);
```

4. Write a method named `secondHalfLetters` that accepts a string as its parameter and returns an integer representing how many of letters in the string come from the second half of the alphabet (that is, have values of `'n'` through `'z'` inclusive). Compare case-insensitively, such that uppercase values of `'N'` through `'Z'` also count. For example, the call `secondHalfLetters("ruminates")`should return `5` because the `'r'`, `'u'`, `'n'`, `'t'`, and `'s'` come from the second half of the alphabet. You may assume that every character in the string is a letter.
5. Write a method named `pow` that accepts a base and an exponent as parameters and returns the base raised to the given power. For example, the call `pow(3, 4)` returns 3 * 3 * 3 * 3 or 81. Do not use `Math.pow` in your solution. Assume that the base and exponent are non-negative.

6. Write a method called `printTriangleType` that accepts three integer arguments representing the lengths of the sides of a triangle and prints what type of triangle it is. The three types are equilateral, isosceles, and scalene. An equilateral triangle has all three sides the same length, an isosceles triangle has two sides the same length, and a scalene triangle has three sides of different lengths. Here are some example calls to `printTriangleType`:

```
printTriangleType(5, 7, 7);
printTriangleType(6, 6, 6);
printTriangleType(5, 7, 8);
printTriangleType(12, 18, 12);
```

The output produced should be the following:

```
isosceles
equilateral
scalene
isosceles
```

Given the following method:

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

Write the output of each of the following calls.

`mystery(1);` _____

`mystery(6);` _____

`mystery(19);` _____

`mystery(39);` _____

`mystery(74);` _____

7. Write a method called `zeroDigits` that accepts an integer parameter and returns the number of digits in the number that have the value `0`. For example, the call `zeroDigits(5024036)` should return `2`, and `zeroDigits(743)` should return `0`. The call `zeroDigits(0)`should return `1`. You may assume that the integer is non-negative. (We suggest you use a `do/while` loop in your solution.)
8. The following method attempts to examine a number and return whether that number is prime (i.e., has no factors other than `1` and itself). A flag named `prime` is used. However, the `Boolean` logic is not implemented correctly, so the method does not always return the correct answer. In what cases does the method report an incorrect answer? Find the problem and change the code so that it will always return a correct result.

9. Write a method named `makeGuesses` that will guess numbers between 1 and 50 inclusive until it makes a guess of at least 48. It should report each guess and at the end should report the total number of guesses made. Below is a sample execution:

```
guess = 43
guess = 47
guess = 45
guess = 27
guess = 49
total guesses = 5
```

10. Write a method named `allDigitsOdd` that returns whether every digit of a given integer is odd. Your method should return `true` if the number consists entirely of odd digits and false if any of its digits are even. 0, 2, 4, 6, and 8 are even digits, and 1, 3, 5, 7, 9 are odd digits. Your method should work for positive and negative numbers.

For example, here are some calls to your method and their expected results:

Call	Value Returned
`allDigitsOdd(4822116)`	`false`
`allDigitsOdd(135319)`	`true`
`allDigitsOdd(9175293)`	`false`
`allDigitsOdd(-137)`	`true`

11. Write a method named `isAllVowels` that returns whether a `String` consists entirely of vowels (a, e, i, o, or u, case-insensitively). If every character of the `String` is a vowel, your method should return `true`. If any character of the `String` is a non-vowel, your method should return `false`. Your method should return `true` if passed the empty string, since it does not contain any non-vowel characters.

For example, here are some calls to your method and their expected results:

Call	Value Returned
`isAllVowels("eIEiO")`	`true`
`isAllVowels("oink")`	`false`

12. Write a method that accepts an array and shits the element of the array to the right/left by 1. You must make changes to the original array. Therefore you cannot make a copy of the array.
13. The following program produces 4 lines of output. Write each line of output below as it would appear on the console.

```
import java.util.*;   // for Arrays class

public class ReferenceMystery1 {
    public static void main(String[] args) {
```

```
        int x = 0;

        int[] a = new int[4];

         x++;



        mystery(x, a);

            System.out.println(x + " " + Arrays.toString(a));


        x++;

        mystery(x, a);

            System.out.println(x + " " + Arrays.toString(a));

    }


        public static void mystery(int x, int[] a) {
            x++;
            a[x]++;
           System.out.println(x + " " + Arrays.toString(a));
        }
    }
```

line 1 [ ]

line 2 [ ]

line 3 [ ]

line 4 [ ]

14. Write a method `averageLength` of code that computes and returns the average `String` length of the elements of an array of `Strings`. For example, if the array contains `{"belt", "hat", "jelly", "bubble gum"}`, the average length returned should be `5.5`. Assume that the array has at least one element.

15. Write a method called `countInRange` that accepts an array of integers, a minimum value, and a maximum value as parameters and returns the count of how many elements from the array fall between the minimum and maximum (inclusive).

For example, in the array `{14, 1, 22, 17, 36, 7, -43, 5}`, there are four elements whose values fall between `4` and `17`.

16. Write a method named `isUnique` that takes an array of integers as a parameter and that returns a boolean value indicating whether or not the values in the array are unique (`true` for yes, `false` for no). The values in the list are considered unique if there is no pair of values that are equal. For example, if a variable called `list` stores the following values:

```
int[] list = {3, 8, 12, 2, 9, 17, 43, -8, 46, 203, 14, 97, 10, 4};
```

    Then the call of `isUnique(list)` should return `true` because there are no duplicated values in this list. If instead the list stored these values:

```
int[] list = {4, 7, 2, 3, 9, 12, -47, -19, 308, 3, 74};
```

    Then the call should return `false` because the value 3 appears twice in this list. Notice that given this definition, a list of 0 or 1 elements would be considered unique.

17. Write a method `priceIsRight` that accepts an array of integers *bids* and an integer *price* as parameters. The method returns the element in the *bids* array that is closest in value to *price* without being larger than *price*. For example, if *bids* stores the elements`{200, 300, 250, 999, 40}`, then `priceIsRight(bids, 280)` should return 250, since 250 is the bid closest to 280 without going over 280. If all bids are larger than *price*, then your method should return `-1`.
    The following table shows some calls to your method and their expected results:

Arrays	**Returned Value**
`int[] a1 = {900, 885, 989, 1};`	`priceIsRight(a1, 800)` returns 1
`int[] a2 = {200};`	`priceIsRight(a2, 120)` returns -1
`int[] a3 = {500, 300, 241, 99, 501};`	`priceIsRight(a3, 50)` returns -1

    You may assume there is at least 1 element in the array, and you may assume that the *price* and the values in *bids* will all be greater than or equal to 1. Do not modify the contents of the array passed to your method as a parameter.