Exam Number:

Points off	1	2	3	4	5	6		Total off	Net Score

CS 305j – Midterm 2 – Fall 2008

Your Name_____

Your UTEID _____

Circle you TA's name: Ann Alex

Instructions:

1. Please turn off your cell phones
2. There are 6 questions on this test.
3. You have 60 minutes to complete the test.
4. You may not use a calculator.
5. Please make your answers legible.
6. When code is required, write Java code.

0. Extra Credit. If you had an account with the user name "student" then the

password would be: _____

1. Simulation. (4 points each, 12 points total.) You are to simulate a method that manipulates an array of integers. Consider the following method:

```
public static void mystery(int[] data){
    int sum = 0;
    int limit = data.length - 3;
    for(int i = data.length - 1; i >= limit; i--){
        sum += data[i];
    }
    System.out.println( sum );
}
```

In the left hand column below is an indication of the values initially in the array of integers named data. The left most element is at position 0. In the right hand column indicate what the output is when method `mystery` is called

`data`	output of method `mystery`
`{0, 0, 0}`	_____
`{0, 1, 2, 3}`	_____
`{4 ,2, 2, 1, 2, 1}`	_____

2. Boolean expressions. (15 points) Write a method to determine the number of points a ticket in a game is worth.

- Each ticket has three integers named a, b, and c.
- If all three integers are the same then the ticket is worth 20 points
- If any two of the integers are the same but the third is different the ticket is worth 10 points.
- If all three integers are different then the ticket is worth the sum of the 3 integers

Here are some examples of call to `points` and what value should be returned:

```
points(0, 0, 0) -> 20
points(0, 0, 1) -> 10
points(2, 4, 6) -> 12 (the sum of 2, 4, and 6)
points(-1, -2, -3) -> -6 (the sum of -1, -2, -3)
points(-10, -2, -2) -> 10
points(1, 5, 1) -> 10
```

Complete the method below:

```
public static int points(int a, int b, int c){
```

3. Array Programming. (20 points) Complete a method that determines how many elements in an array of `int`s are greater than or equal to a given value.

```
public static int numGreater(int[] data, int value)
```

Here are some examples of expected return values for various arrays and values.

```
numGreater( {}, 10 ) -> 0
numGreater( {1, 2, -4, 1, 6}, 10 ) -> 0
numGreater( {1, 20, -4, 10, 6}, 10 ) -> 2
numGreater( {1, 2, -4, 1, 6}, 0 ) -> 4
numGreater( {1, 2, -4, 1, 6}, -10 ) -> 5
```

Complete the method below:

```
public static int numGreater(int[] data, int value){
```

4. Array Programming. (15 points) Complete a method that given an array of `doubles` returns `true` if the elements of the array are sorted in ascending order, `false` otherwise.

Here are some examples of expected return values for various arrays.

```
isSorted( {} ) -> true
isSorted( {1.0} ) -> true
isSorted( {0.0, 0.0, 0.0, 0.0} ) -> true
isSorted( {0.0, 0.1, 0.0, 0.0}  ) -> false
isSorted( {0.5, 1.0, 1.0, 2.5}  ) -> true
isSorted( {0.5, 1.0, 1.0, 0.5}  ) -> false
```

Complete the following method:

```
public static boolean isSorted(double[] nums){
```

5. Programming. (20 points) Complete a method that plays the game "Fizz Bang".

- Fizz Bang is a simple counting game.
- Given a starting number and an ending number print out the numbers in that range with the following exceptions.
- Numbers that are multiples of 5 and 3 cause "FizzBang" to be printed out.
- Numbers that are just multiples of 3 cause "Fizz" to be printed out.
- Numbers that are just multiples of 5 cause "Bang" to be printed out.

If the starting number is 1 and the ending number is 20 here is the expected output.

```
1 2 Fizz 4 Bang Fizz 7 8 Fizz Bang 11 Fizz 13 14 FizzBang 16 17 Fizz 19 Bang
```

Complete the following method:

```
// start and stop will both be greater than 0.
// start will be <= stop. All output will appear on one line.
public static void fizzBang(int start, int stop){
```

6. Program Logic (18 points) Consider the following method. For each of the six points labeled by comments and each of the three assertions in the table, write whether the assertion is ***always*** true, ***sometimes*** true, or ***never*** true at that point in the code. Abbreviate ***always*** with an A, ***sometimes*** with an S and ***never*** with an N.

```
public static int assertionPractice(int num, int target){
      int count = 0;
      int temp = 0;
      if( num > 0 ){
            // Point A

            while( num != 0 ){
                  // Point B

                  temp = num % 10;

                  if( temp == target ){
                        count++;
                        // Point C
                  }

                  num = num / 10;
                  // Point D
            }
            // Point E
      }

      // Point F
      return count;
}
```

Abbreviate ***always*** with an A, ***sometimes*** with an S and ***never*** with an N.

	num != 0	count > 0	temp == target
Point A			
Point B			
Point C			
Point D			
Point E			
Point F			

Scratch Paper