Exam Number:

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

CS 305j – Midterm 2 – Spring 2010

Your Name___________________________________

Your UTEID __________________________________

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.

1. Guest Lecture. (8 points) Professor Calvin Lin of the UT Computer Science Department gave a lecture on Why Programming is Important.

A. Professor Lin presented the anecdote that the microchip in a musical greeting card you can buy today for five dollars has more computing power than the early computers of the 1940s and 1950s. *Moore's Law* describes one of the trends that made this possible. **Briefly** (in a sentence or two) explain what Moore's Law is:

B. Professor Lin argued that computation and programming enhance the search for new medicinal drugs. **Briefly** (in a sentence or two) explain how computation and programming enhance the search for new medicinal drugs:

2. Simulation. (16 points) What is output when the following program is run?

```
public class Midterm2{

     public static void main(String[] args){
          int x = 5;
          int y = 12;
          methodA(y, x);
          System.out.println("Line 2: " + x + " " + y);

          int[] list = {5, 3, 7};
          System.out.println("Line 3: " + list[0] + " " + list[2]);
          methodB(list);
          System.out.println("Line 5: " + list[0] + " " + list[2]);

          list = new int[5];
          System.out.println("Line 6: " + list[1] + " " + list.length);
          methodC(list);
          System.out.println("Line 8: " + list[0] + " " + list[2]);
     }

     public static void methodA(int x, int y){
          x++;
          y--;
          System.out.println("Line 1: " + x + " " + y);
     }

     public static void methodB(int[] data){
          data[0]++;
          data[2] = 9;
          System.out.println("Line 4: " + data[0] + " " + data[2]);
     }

     public static void methodC(int[] values){
          values[0]++;
          values = new int[]{12, 14, 19, 17};
          System.out.println("Line 7: " + values[0] + " " + values[2]);
     }
}
```

Complete the output of the program below:

```
Line 1:
Line 2:
Line 3:
Line 4:
Line 5:
Line 6:
Line 7:
Line 8:
```

3. Method that return values. (18 points) Write a method to determine if two lines are parallel. Each line will be designated with 4 integers. The integers will represent the x and y coordinates of 2 distinct points on each line. Recall lines are parallel if they have the same slope. Slope is the ratio of the change in the y values between two points on a line and the change in x values of those same two points.

Here are some example calls and their expected results

```
areParallel(1, 3, 3, 6, 5, 10, 9, 16) -> true
areParallel(1, 3, 3, 6, 5, 10, 8, 16) -> false
areParallel(0, 5, 0 , 10, 5, 3, 10, 3) -> false
areParallel(0, 5, 0 , 10, 7, 3, 7, 10) -> true
```

Complete the method below:

```
/* Determine if two lines are parallel or not.

   x1, y1 and x2, y2 represent two distinct points on line 1.
   x3, y3 and x4, y4 represent two distinct points on line 2.

   return true if the first line is parallel with the second line, false
   otherwise.
*/

public static boolean areParallel(int x1, int y1, int x2, int y2,
                             int x3, int y3, int x4, int y4)
{
```

4. Array Programming. (30 points) Complete a method that given an array of doubles returns the difference between the mean of the elements in the array and the median of the elements in the array. Recall the mean of a group of values is the average of those values. The median of a group of values is the middle element in the group assuming the values are sorted. You may assume the elements in the array are already sorted in ascending order and that the length of the array is greater than 0.

If a sorted array has an odd number of values the middle element is the median:

```
{-2.5, 0.0, 3.5, 12.5, 27.0} -> median is 3
```

If a sorted array has an even number of values the median is the average of the two middle values:

```
{-2.5, 0.0, 3.5, 12.5, 27.0, 27.1} -> median is (3.5 + 12.5) / 2 = 8.0
```

Write a method that returns the mean of the elements in an array minus the median of the elements in the array.

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

```
diffMeanAndMedian( {-2.5, 0.0, 3.5, 12.5, 27.0} ) -> 4.6
diffMeanAndMedian({-2.5, 0.0, 3.5, 12.5, 27.0, 27.1} ) -> 3.266666666666667
diffMeanAndMedian ( {2.5} ) -> 0
diffMeanAndMedian ( {-5.0, -5.0, -5.0, -5.0} ) -> 0
diffMeanAndMedian ( {1.0, 1.0, 1.0, 1.0, 100.0} ) -> 19.8
```

Complete the following method:

```
public static double diffMeanAndMedian (double[] data)
{
```

5. Understanding Variables (10 points) Consider the following program. Draw a picture of all the variables that exist and the values they hold when the program is run and the comment marked `// WHAT VARIABLES??` is reached. Include all variables that exist in the program except `args`, not just the variables that are in scope in method `aToyMethod`.

```
public class Vars
{
     public static void main(String[] args)
     {
          int x = 2;
          int y = x * 2;
          int[] data = {12, x, 3, 4};
          aToyMethod(data, x, y);
     }

     public static int aToyMethod(int[] list, int a, int b)
     {
          int x = list[1] * b;
          // WHAT VARIABLES??

          return list[0] * list[list.length - 1];
     }
}
```

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(String st, char tgt, int nth){
      int result = -1;
      int count = 1;
      int i = 0;
      // Point A
      if( (st.length() > 0) && (st.charAt(i) == tgt) && (nth > 0) )
      {
            // Point B
            while( count < nth && i < st.length() )
            {
                  // Point C
                  if( st.charAt(i) == tgt )
                  {
                        count++;
                        // Point D
                  }
                  i++;
            }
            if( count == nth )
            {
                  result = i;
                  // Point E
            }

      }
      // Point F
      return result;
}
```

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

	`st.charAt(i) == tgt`	`count == nth`	`result == -1`
Point A			
Point B			
Point C			
Point D			
Point E			
Point F			

Scratch Paper

Scratch Paper

Scratch Paper