CSCI 135 Exam #1 **Name: ___________________________________**
Fundamentals of Computer Science I
Fall 2014

This exam consists of 8 problems on the following 8 pages.

You may use your two-sided hand-written 8 ½ x 11 note sheet during the exam. No calculators, computers, or communication devices of any kind are permitted.

If you have a question, raise your hand and I will stop by. Since partial credit is possible, **please write legibly and show your work**.

Problem	**Points**	**Score**
1	**6**	
2	**12**	
3	**12**	
4	**6**	
5	**3**	
6	**10**	
7	**10**	
8	**6**	
Total	**65**	

1. Loops, input (6 points). Consider the following program:

```
public class Prob1
{
    public static void main(String [] args)
    {
        int val = Integer.parseInt(args[0]);
        String s1 = args[1];
        String s2 = args[2];

        for (int i = 3; i < args.length; i++)
        {
            if (Integer.parseInt(args[i]) < val)
                System.out.print(s1);
            else
                System.out.print(s2);
        }
        System.out.println("!");
    }
}
```

Give the output of `Prob1` for each of the following commands. If the program would crash, write "error":

Command	**Program output**
`% java Prob1 10 a`	error
`% java Prob1 -10 a b`	!
`% java Prob1 10 a b 20 5`	ba!
`% java Prob1 3.3 a b 5`	error
`% java Prob1 3 a b 0 1 2`	aaa!
`% java Prob1 1000 -1 1 500 1000`	-11!

2. Code fragments (12 points).

a) Give a code fragment that uses a loop to print out the even numbers from 1 to 100 (inclusive, i.e. 2, 4, 6, …, 100). Each number should be output on its own line.

```
for (int i = 2; i <= 100; i = i + 2)
    System.out.println(i);
```

b) Give a code fragment that declares and instantiates an array of 10 `double` values.

```
double [] d = new double[10];
```

c) Using the array from part b, give a code fragment that sets each element of the array to a different random value in [0.0, 1.0). For full credit, your fragment should work (without change) even if the size of the array in part b is changed.

```
for (int i = 0; i < d.length; i++)
    d[i] = Math.random();
```

d) Give a static method `avg2` that takes two `int` values as arguments and returns the `double` value of the average of the two numbers. For example, `avg2(3, 4)` would return `3.5`. For full credit, be sure to include an access modifier that lets code in any class call the `avg2()` method.

```
public static double avg2(int a, int b)
{
    return (a + b) / 2.0;
}
```

3. Java keywords (12 points). Match each of the Java keywords on the left with the letter of the ***best*** description of a potential use of the keyword. ***Not all letters will be used, a letter can only be used once***.

`boolean` __B__	A. Used in a method declaration to signify the method has no return result.
`break` __L__	B. A primitive data type that can only take on two distinct values.
`final` __G__	C. A primitive data type that can hold a floating point value.
`new` __E__	D. A reference data type that can only take on two distinct values.
`null` __J__	E. Causes a reference data type to be instantiated including calling the relevant constructor.
`private` __F__	F. Used to achieve data encapsulation of instance variables.
`public` __H__	G. Used to prevent a variable from having its value changed after its initial assignment.
`return` __R__	H. Ensures that a method or instance variable is visible to programs in any other class.
`static` __P__	I. Used to specify a variable name refers to an instance variable and not a local variable.
`this` __I__	J. The value of a reference variable before it has been instantiated using the `new` operator.
`do` __T__	K. The value of a primitive variable after dividing by zero.
`void` __A__	L. Causes the enclosing loop to terminate without executing any further statements in the body of the loop.
	M. Causes the enclosing loop to terminate after executing the remaining statements in the body of the loop.
	N. Signifies a method that is forbidden from calling itself recursively.
	O. Signifies a method that is overloading an existing method.
	P. Signifies a method that can be called by preceding the method name with the name of the class in which the method appears (e.g. `MyClass.foo`).
	Q. Causes the entire Java program to immediately exit.
	R. Used to specify the output value (if any) returned by a method. No further lines of code in the method are executed after hitting this keyword.
	S. Start of a loop construct in which the body executes a minimum of 0 times.
	T. Start of a loop construct in which the body executes a minimum of 1 time.

4. Recursion (6 points). Consider the following recursive method:

```
public static void foo(int n)
{
    if (n <= 0)
        return;
    System.out.println(n);
    foo(n - 3);
    foo(n - 2);
}
```

For each of the following calls, give the sequence of integers printed. If the program would produce no output, write "no output". If the program would crash with a stack overflow, write "stack overflow".

a) `foo(-1)`

no ouput

b) `foo(2)`

2

c) `foo(6)`

6
3
1
4
1
2

5. Debugging (3 points). The following data type represents a point in 3-dimensional space:

```
public class Point3D
{
    private double x = 0.0;
    private double y = 0.0;
    private double z = 0.0;

    public Point3D(double x, double y, double z)
    {
          this.x = x;
          this.y = y;
          this.z = z;
    }

    public String toString()
    {
        return String.format("(%.2f, %.2f, %.2f)", x, y, z);
    }

    public double distance(Point3D other)
    {
        double deltaX = x - other.x;
        double deltaY = y - other.y;
        double deltaZ = z - other.z;

        return Math.sqrt(deltaX*deltaX + deltaY*deltaY + deltaZ*deltaZ);
    }

    public static void main(String [] args)
    {
        Point3D p = new Point3D(1, 2, 3);
        System.out.println(p.toString());
    }
}
```

Running the test main() method currently produces the following:

```
% java Point3D
(0.00, 0.00, 0.00)
```

Correct the line(s) in the above program such that it produces the expected output:

```
% java Point3D
(1.00, 2.00, 3.00)
```

6. Arrays, objects (10 points). Assume the problem in `Point3D` has been corrected and all method work correctly. You are working on a client program `AllDiff` that reads in a points from standard input into an array. Rather than specifying the number of points via standard input, instead the number of points is specified as the first command line argument to the program. The program outputs the distance ***rounded to two decimal places*** between all pairs of points. Here is an example input file and the desired output format:

```
% more 3points.txt
0.0 0.0 0.0
1.0 1.0 1.0
0.5 1.5 2.0

% java AllDiff 3 < 3points.txt
(0.00, 0.00, 0.00) <-> (1.00, 1.00, 1.00) = 1.73
(0.00, 0.00, 0.00) <-> (0.50, 1.50, 2.00) = 2.55
(1.00, 1.00, 1.00) <-> (0.50, 1.50, 2.00) = 1.22
```

Fill in the missing code in the following program such that `AllDiff` works as described.

```
public class AllDiff
{
    public static void main(String[] args)
    {
        final int N = Integer.parseInt(args[0]);

        Point3D [] points = new Point3D[N];
        for (int i = 0; i < N; i++)
        {
            points[i] = new Point3D(StdIn.readDouble(),
                                    StdIn.readDouble(),
                                    StdIn.readDouble());
        }

        for (int i = 0; i < N; i++)
        {
            for (int j = i + 1; j < N; j++)
            {
                double dist = points[i].distance(points[j]);
                System.out.printf("%s <-> %s = %.2f\n",
                                  points[i].toString(),
                                  points[j].toString(),
                                  dist);
            }
        }
    }
}
```

7. Algorithms, objects (10 points). You are working on a different client program `PointStats` that makes use of your `Point3D` data type. This program's job is to read in via standard input a sequence of 0 or more points and computes:

- The distance of the path connecting the points in the sequence (the path goes in order, i.e. going from the first point in the input, to the second point, and so on).
- The minimum distance between any two consecutive points in the sequence.
- The maximum distance between any two consecutive points in the sequence.

Here is an example run with four points lying on the x-axis:

```
% more 4points.txt
0.0  0.0  0.0
1.0  0.0  0.0
3.0  0.0  0.0
2.5  0.0  0.0
```

```
% java PointStats < 4points.txt
Distance = 3.5
Min = 0.5
Max = 2.0
```

Write the letters that combine to create a correct implementation. ***Letters may be used 0 or more times.***

```
public class PointStats
{
  public static void main(String[] args)
  {
    Point3D last = null;
    double  sum  = 0.0;

    double  min  = __G__;
    double  max  = __H__;
    while (__N__)
    {
      Point3D p = __K__  __J__ (StdIn.readDouble(),
                                StdIn.readDouble(),
                                StdIn.readDouble());

      if (__S__)
      {
        double dist = __C__;
        sum = sum + dist;

        min = __V__;

        max = __W__;
      }
      last = __E__;
    }
    System.out.println("Distance = " + sum);
    System.out.println("Min = " + min);
    System.out.println("Max = " + max);
  }
}
```

```
A. Math.abs(last - p)
B. last = dist
C. last.distance(p)
D. last
E. p
F. last++
G. Double.POSITIVE_INFINITY
H. Double.NEGATIVE_INFINITY
I. 0.0
J. Point3D
K. new
L. break
M. final
N. !StdIn.isEmpty()
O. StdIn.isEmpty()
P. StdIn.readDouble()
Q. last < max
R. last == null
S. last != null
T. Math.min(dist, sum)
U. Math.max(dist, sum)
V. Math.min(dist, min)
W. Math.max(dist, max)
```

8. Designing objects (6 points).

You decide your `Point3D` data type also needs a text label.

a) Give the declaration of an appropriate instance variable to support this label feature. For full credit, demonstrate the principle of data encapsulation.

Add a private instance variable to hold a string. For example:

```
private String label;
```

b) Describe how you would change the class in order to support setting of the label during the creation of the `Point3D` object.

Add an additional parameter to the constructor that allows setting of the label instance variable. For example:

```
public Point3D(double x, double y, double z, String label)
{
    this.x = x;
    this.y = y;
    this.z = z;
    this.label = label;
}
```