CSCI 135 Exam #2 **Name: _____**
Fundamentals of Computer Science I
Fall 2013

This exam consists of 6 problems on the following 6 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	10	
2	10	
3	14	
4	10	
5	12	
6	12	
Total	68	

1. Loops, input (10 points). Consider the following program:

```
public class Prob1
{
    public static void main(String [] args)
    {
        for (int i = 0; i < args.length; i++)
        {
            System.out.print(args[i]);
            if (i < args.length - 1)
                System.out.print(", ");
        }
        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 hi ho hi ho`	`hi, ho, hi, ho.`
`% java Prob1 hello`	`hello.`
`% java Prob1`	`.`
`% java Prob1 one two three`	`one, two, three`
`% java Prob1 1 2 3`	`1, 2, 3.`

2. Multiple choice (10 points, 2 points each). Circle the ***best*** single answer.

I. Data encapsulation is a desirable goal when implementing your own data types. Which of the following is the ***best way*** to achieve data encapsulation:

a) Declare all methods using lower camel case.
b) Declare all methods as `static`.
c) Declare all instance variables as `final`.
d) Declare all instance variables as `private`.

II. Consider the following code fragment:

```
Point p1 = new Point(1.0, 1.0);
Point p2 = new Point(2.0, 2.0);
p1 = p2;
```

The next time the Java garbage collector runs, what point(s) will be garbage collected?

a) No points will be garbage collected.
b) The point that was at (1.0, 1.0).
c) The point that was at (2.0, 2.0).
d) Both the point at (1.0, 1.0) and the point at (2.0, 2.0).

III. You are debugging a recursive method that currently produces a `StackOverflowError` exception. What is the ***most likely*** explanation of the cause of the problem?

a) You forgot to include a `toString()` method in the enclosing class.
b) You forgot to include a base case in your recursive method.
c) You forgot to use the `this` keyword when referring to parameters to the recursive method.
d) You forgot to use the `final` keyword when referring to parameters to the recursive method.

IV. The class `MyHelper` has a static method declared as follows:

```
public static void foo(int a)  { /* some code */ }
```

Which of the following lines would be a valid way to call the above method from the `main()` method of the class `MyGame`:

a) `MyHelper.foo(4 + 5);`
b) `MyHelper.foo(4.5);`
c) `foo(4 + 5);`
d) `MyGame.foo(4);`

V. Queues and stacks are two fundamental abstract data types (ADTs) in computer science. Which of the following statements is ***TRUE***:

a) Queues implement a last-in first-out (LIFO) removal policy.
b) Stacks implement a first-in first-out (FIFO) removal policy.
c) Traditionally, a queue ADT names the add operation `pop()` and the remove operation `push()`.
d) Queues and stacks can be implemented using a linked list data structure.

3. Variables and expressions (14 points).

a) Give the type and value of each of the following expressions. If an expression causes an error, write "error" in the type column (and leave the value blank).

Expression	Type	Value
`0.0 * 123`	double	0.0
`123 * (int) 0.1`	int	0
`(Math.random() < 1.0)`	boolean	true
`1 / 2.0 + 2`	double	2.5
`11 % 2`	integer	1
`Integer.parseInt("12" + "34")`	integer	1234
`(2 <= 3) || (3 < 2)`	boolean	true

b) Consider the following class:

```
01 public class Scope
02 {
03     private int a;
04     public final double PI = 3.0;
05
06     public void foo(int a, String [] s)
07     {
08
09         for (int i = 0; i < s.length; i++)
10         {
11
12         }
13
14         double d = 2.5;
15
16     }
17
18     public void foo2(Scope other)
19     {
20
21     }
22 }
```

For each of the specified lines in the above program, check the boxes of the variables that would be in scope at that line (i.e. you could use the variable name in some appropriate way at that line and it would compile).

Line	a	PI	this.a	d	other.a	s	i
08	X	X	X			X	
11	X	X	X			X	X
15	X	X	X	X		X	
20	X	X	X		X		

4. Using objects (10 points). Here is the API for a class that represents charged particles:

```
public class Charge
---------------------------------------------------------------------------
       Charge(double x0, double y0, double q0) // create particle at (x0,y0) with charge q0
double potentialAt(double x, double y)         // electric potential at (x,y) due to charge
String toString()                              // string representation of particle
```

You are writing a client program FourCharge that takes a double value w from the first command line argument, creating four Charge objects each with charge value 1.0 that are each distance w in each of the four cardinal directions from (0.5, 0.5). It also prints the potential at (0.25, 0.5) due to all the charged particles. Fill in the missing code (denoted by underscores) in the following program:

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

   double w = Double.parseDouble(args[0]);

   Charge c1 = new Charge(0.5 + w, 0.5, 1.0);

   Charge c2 = new Charge(0.5 - w, 0.5, 1.0);

   Charge c3 = new Charge(0.5, 0.5 + w, 1.0);

   Charge c4 = new Charge(0.5, 0.5 - w, 1.0);

   double x = 0.25;
   double y = 0.5;


   double v1 = c1.potentialAt(x, y);

   double v2 = c2.potentialAt(x, y);

   double v3 = c3.potentialAt(x, y);

   double v4 = c4.potentialAt(x, y);


   System.out.println(v1 + v2 + v3 + v4);

}
```

5. Arrays (12 points)

a) Give a single line of Java code that declares and instantiates an array containing 1000 **boolean** values.

```
boolean [] flags = new boolean[100];
```

b) Give Java code that sets each value in the previous array to **true**. For full credit, make your code work regardless of how big the array is.

```
for (int i = 0; i < flags.length; i++)
   flags[i] = true;
```

c) Give a single line of Java code that declares and instantiates an array that can hold 50 Charge objects (see the previous problem for details about the Charge class).

```
Charge [] charges = new Charge[50];
```

d) After the line of code from part c, what value does every element of your Charge array have?

```
null
```

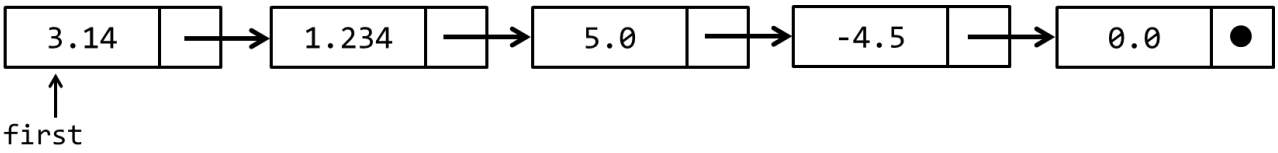
e) Give Java code that fills your Charge array with random particles. Each particle should have a random x-position in [0.0, 1.0), a random y-position in [0.0, 1.0), and each particle has a 50% chance of having a negative charge of -1.0 and a 50% chance of having a positive charge of +1.0.

```
for (int i = 0; i < charges.length; i++)
{
   if (Math.random() < 0.5)
      charges[i] = new Charge(Math.random(), Math.random(), -1.0);
   else
      charges[i] = new Charge(Math.random(), Math.random(), +1.0);
}
```

6. Linked lists (12 points). You have developed a linked list class that holds a sequence of **double** values. It uses the following inner class Node:

```
private class Node
{
    double value;
    Node next;
}
```

Currently the linked list has the following data and structure:



Below are some instance methods contained in your linked list class. Your class has an instance variable `first` that points to the front of the linked list. In each method's box, write the letter that ***best*** describes what that method does with respect to the linked list. Each letter will be used ***at most once***.

```
public int foo1()
{
  int a = 0;
  for (Node current = first;
       current != null;
       current = current.next)
  {
    a++;
  }
  return a;
}
```
C

```
public void foo2()
{
  first = null;
}
```
I

```
public double foo3()
{
  double a = Double.NEGATIVE_INFINITY;
  for (Node current = first;
       current != null;
       current = current.next)
  {
    if (current.value > a)
      a = current.value;
  }
  return a;
}
```
B

```
public double foo4()
{
  double a = 0.0;
  for (Node current = first;
       current != null;
       current = current.next)
  {
    a = current.value;
  }
  return a;
}
```
D

```
public void foo5()
{
  if (first != null)
    first = first.next;
}
```
J

```
public double foo6()
{
  double a = 0.0;
  for (Node current = first;
       current != null;
       current = current.next)
  {
    a = a + current.value;
  }
  return a;
}
```
G

A. Returns the minimum value in the list.
B. Returns the maximum value in the list.
C. Returns the number of entries in the list.
D. Returns the value of the last entry in the list.
E. Returns sum of all positive numbers in the list.
F. Returns sum of all negative numbers in the list.
G. Returns sum of all numbers in the list.
H. Deletes the last entry in the list.
I. Deletes all entries in the list.
J. Delete the first entry in the list.
K. Deletes all entries that are equal to 0.0.
L. Inserts a new entry at the beginning of the list.
M. Inserts a new entry at the end of the list.
N. Creates a new list with a single entry.