CSCI 135 Exam #0 **Name: ___________________________________**
Fundamentals of Computer Science I
Fall 2012

This exam consists of 7 problems on the following 6 pages.

You may use your single-side 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	**8**	
2	**9**	
3	**6**	
4	**4**	
5	**14**	
6	**10**	
7	**8**	
Total	**59**	

1. Loops (8 points). Consider the following code fragment:

```
int i = -7;
while (i < 7)
{
   System.out.println(i);
   i = i + 3;
}
```

Give the output produced by the code.

Convert the above code fragment to use a `for` loop instead of a `while` loop.

2. Input and conditionals (9 points). You are writing a program for an airline that determines whether a checked-bag of a given length, width, height, and weight is allowed on the aircraft. The ***sum*** of a bag's length, width and height must be ***less than or equal to 62 inches***. In addition, the bag's weight must be ***less than or equal to 50 pounds***

Your program takes exactly four ***command-line arguments*** for the length, width, height, and weight (in that order). Assume all input may be specified to several decimal points of precision (e.g. length = 10.3, width = 46.5, height = 13.0, weight = 44.74). The program prints "Good bag" if a bag meets the size and weight restrictions and "Bad bag" if it does not. Below is the skeleton of the program. In the empty boxes, write the letter of the code fragments that combine to create a correct implementation.

```
public class CheckedBag
{
    
    {
        


        boolean bad = false;

           bad = true;

        if (bad) System.out.println("Bad bag");
        else     System.out.println("Good bag");
    }
}
```

A. `public static void main(String[] args)`	**B.** `public static void main(double[] args)`
C. `public static void main()`	**D.** `void main(String[] args)`
E. `public static void check(String[] args)`	**F.** `public static void main(String args)`

G. `int length = Integer.parseInt(args[0]);`
`int width  = Integer.parseInt(args[1]);`
`int height = Integer.parseInt(args[2]);`
`int weight = Integer.parseInt(args[3]);`
	H. `double length = Double.parseDouble(args[1]);`
	`double width  = Double.parseDouble(args[2]);`
	`double height = Double.parseDouble(args[3]);`
	`double weight = Double.parseDouble(args[4]);`

I. `double length = args[0];`
`double width  = args[1];`
`double height = args[2];`
`double weight = args[3];`
	J. `double length = Double.parseDouble(args[0]);`
	`double width  = Double.parseDouble(args[1]);`
	`double height = Double.parseDouble(args[2]);`
	`double weight = Double.parseDouble(args[3]);`

K. `double length = StdIn.readDouble();`
`double width  = StdIn.readDouble();`
`double height = StdIn.readDouble();`
`double weight = StdIn.readDouble();`
	L. `double length = StdIn.readDouble(args[0]);`
	`double width  = StdIn.readDouble(args[0]);`
	`double height = StdIn.readDouble(args[0]);`
	`double weight = StdIn.readDouble(args[0]);`

M. `if ((length + width + height <= 62.0) ||` `    (weight <= 50.0))`	**N.** `if ((length + width + height <= 62) &&` `    (weight <= 50))`
O. `if (length > 62 || width > 62 ||` `    height > 62 || weight > 50)`	**P.** `while ((length + width + height > 62) &&` `       (weight > 50))`
Q. `if ((length + width + height > 62) ||` `    (weight > 50))`	**R.** `if ((length + width + height > 62) &&` `    (weight > 50))`

3. Arrays (6 points). What values does the following code put in the array a[]?

```
int N = 10;
int[] a = new int[N];
a[0] = 1;
a[1] = 1;
for (int i = 2; i < N; i++)
   a[i] = a[i-1] + a[i-2];
```

Complete the following table showing the array elements and values. The first row has been done for you:

Array element	Value
a[0]	1

4. Arrays and methods (4 points). The following methods are supposed to square the values in the passed in array of double values, storing the new squared value in the original array. Circle the ***only*** method that does its job correctly for ***any*** passed in double array.

```
public static void squareA(double [] a)
{
   for (int i = 0; i < 100; i++)
      a[i] = a[i] * a[i];
}
```

```
public static void squareC(double [] a)
{
   int i = a.length;
   while (i >= 1)
   {
      i = i - 1;
      a[i] = a[i] * a[i];
   }
}
```

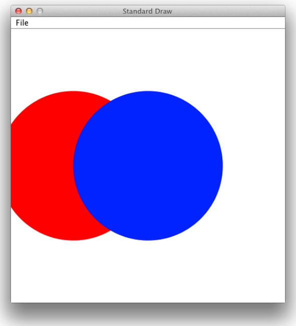
```
public static void squareB(double [] a)
{
   for (int i = 0; i <= a.length; i++)
      a[i] = a[i] * a[i];
}
```

```
public static void squareD(double [] a)
{
   for (int i = a.length - 1; i > 0; i--)
      a[i] = a[i] * a[i];
}
```

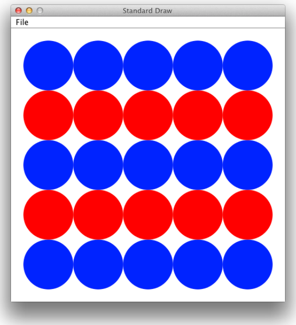
5. Java expressions (14 points). Give the data type and value of each of the following expressions. If an expression causes an error, write "error" in the type column and leave the value blank.

Expression	Type	Value
`6 + 2`		
`6.0 + 2`		
`(int) 6.0 + 2`		
`(int) (10 + 0.9)`		
`Integer.parseInt("1234")`		
`Integer.parseInt("9.99")`		
`Double.parseDouble("23.1%")`		
`"" + 42`		
`2 * Math.max(1.0, 2.0)`		
`5 / 10`		
`5 / (double) 10`		
`10 % 5`		
`((5 != 0) && (5 <= 3))`		
`((5 != 0) || (5 <= 3))`		

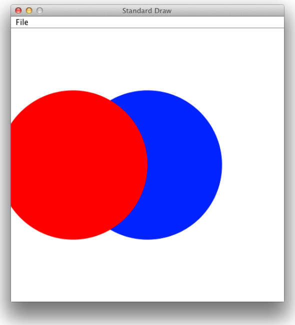
6. Drawing (10 points). On the left are the output from a number of programs. Label each output with the letter of the program that generated it (each letter is used exactly once). All programs used the default `StdDraw` coordinates with (0.0, 0.0) being the lower-left corner and (1.0, 1.0) being the upper-right corner.



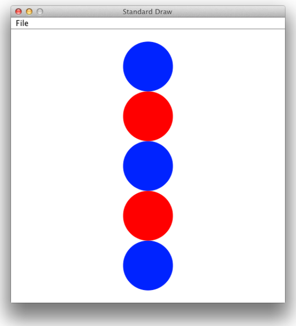
________ Program



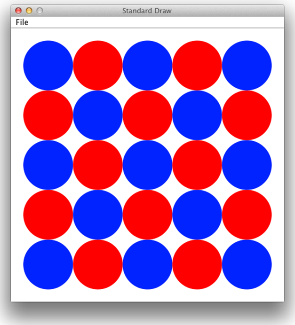
________ Program



________ Program



________ Program



________ Program

Program A

```
StdDraw.setPenColor(StdDraw.RED);
StdDraw.filledCircle(0.2, 0.5, 0.3);
StdDraw.setPenColor(StdDraw.BLUE);
StdDraw.filledCircle(0.5, 0.5, 0.3);
```

Program B

```
StdDraw.setPenColor(StdDraw.BLUE);
StdDraw.filledCircle(0.5, 0.5, 0.3);
StdDraw.setPenColor(StdDraw.RED);
StdDraw.filledCircle(0.2, 0.5, 0.3);
```

Program C

```
final int N = 5;
double inc = 1.0 / N;
for (int x = 0; x < N; x++) {
   for (int y = 0; y < N; y++) {
      StdDraw.setPenColor(StdDraw.RED);
      if (y % 2 == 0)
         StdDraw.setPenColor(StdDraw.BLUE);
      StdDraw.filledCircle((x + 0.5) * inc,
                           (y + 0.5) * inc,
                           inc / 2.0);
   }
}
```

Program D

```
final int N = 5;
double inc = 1.0 / N;
for (int x = 0; x < N; x++) {
   for (int y = 0; y < N; y++) {
      StdDraw.setPenColor(StdDraw.RED);
      if ((x + y) % 2 == 0)
         StdDraw.setPenColor(StdDraw.BLUE);
      StdDraw.filledCircle((x + 0.5) * inc,
                           (y + 0.5) * inc,
                           inc / 2.0);
   }
}
```

Program E

```
final int N = 5;
double inc = 1.0 / N;
for (int y = 0; y < N; y++) {
   StdDraw.setPenColor(StdDraw.RED);
   if (y % 2 == 0)
      StdDraw.setPenColor(StdDraw.BLUE);
   StdDraw.filledCircle(0.5,
                        (y + 0.5) * inc,
                        inc / 2.0);
}
```

7. Output, loops and Strings (8 points). The following program reads in two arguments from the command-line, a String pattern p and a positive integer N. The program then prints out N lines of output. The 1st line of output is just the pattern, the 2nd line has twice as many copies of the pattern (2 patterns), the 3rd line has twice again as many (4 patterns), the 4th line has twice again as many (8 patterns), and so on. Here is an example run:

```
% java DoublingPattern ab 5
ab
abab
abababab
abababababababab
abababababababababababababababab
```

Fill in the missing part of the program that performs the output.

```java
public class DoublingPattern
{
    public static void main(String[] args)
    {
        String p = args[0];
        int N = Integer.parseInt(args[1]);









    }
}
```