CSCI 135 Midterm **Name: _____**
Fundamentals of Computer Science I
Fall 2011

This exam consists of 12 problems on the following 11 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	4	
2	14	
3	4	
4	4	
5	9	
6	9	
7	15	
8	6	
9	9	
10	12	
11	15	
12	10	
Total	111	

1. **Loops** (4 points). Given the following `while` loop, convert it into a `for` loop:

```
int i = 0;
while (i < 100)
{
   System.out.println(i);
   i = i + 2;
}
```

```
for (int i = 0; i < 100; i += 2)
{
   System.out.println(i);
}
```

2. **Java expressions** (14 points). 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
`6 * 0.2`	**double**	**1.2**
`(int) 6 * 0.2`	**double**	**1.2**
`6 * (int) 0.2`	**int**	**0**
`(int) (6 * 0.2)`	**int**	**1**
`Integer.parseInt("42")`	**int**	**42**
`Integer.parseInt("42.1")`	**error**	
`Double.parseDouble("$1.23")`	**error**	
`"" + 3.14`	**String**	**"3.14"**
`"hello + " + "world"`	**String**	**"hello + world"**
`10 / 3`	**int**	**3**
`12 / 4.0`	**double**	**3.0**
`10 % 3`	**int**	**1**
`((5 > 0) && (5 < 3))`	**boolean**	**false**
`(1 == (5 * 2 - 9))`	**boolean**	**true**

3. **Arrays** (4 points). Among the following code fragments, circle the letter(s) of those that will **not** cause a compile time error.

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

B. `int a = {1, 2, 3};`

C. `double [] d = double[10];`

D. `int [] a = {1, 2, 3}; int b = a;`

E. `int [] a = new double[10];`

F. `int [10] a = new int[];`

G. `int [] a;`

4. **Variable scope** (4 points). Consider the following program:

```
public class Mystery
{
   public static int mystery(int i)
   {
      return i * i * i;
   }

   public static void main(String [] args)
   {
      for (int i = 1; i <= 1000; i++)
      {
         System.out.println(mystery(i));
      }
   }
}
```

Among the following statements, circle the letter(s) that are **true**:

A. Will not compile because the variable `i` is not declared in the method `mystery()`.

B. Prints only a few lines because of scope clash with variable `i`.

C. Prints the cubes of the integers from 1 to 1000 (nothing more and nothing less).

D. Prints the cubes of the integers from 1 to 999 (nothing more and nothing less).

E. Goes into an infinite loop

F. Will not compile because of name clash on the word "mystery".

5. **Arrays and debugging** (9 points). Consider the following program which is supposed to do the following: read an integer N from standard input, read N strings from standard input, and then print the strings in reverse order.

```
 1 public class ReverseInput
 2 {
 3    public static void main(String [] args)
 4    {
 5       int N = StdIn.readInt();
 6       String s;
 7       for (int i = 1; i < N; i++)
 8          s[i] = StdIn.readString();
 9       for (int i = N; i >= 0; i--)
10          System.out.println(s[i]);
11    }
12 }
```

This program has three bugs.

A. Which bug prevents the program from *compiling* successfully? Identify the line number with the mistake that causes the compile error. Give a correct version of this line of code.

Line number **__6__**

Correct version: **`String [] s = new String[N];`**

B. After fixing the first bug, which bug causes the program to crash with a *runtime error*? Identify the line number causing the runtime error. Give a correct version of this line.

Line number **__9__**

Correct version: **`for (int i = N - 1; i >= 0; i--)`**

C. After fixing the first two bugs, which bug causes the program to produce incorrect output? Identify the line number causing the bug. Give a correct version of this line.

Line number **__7__**

Correct version: **`for (int i = 0; i < N; i++)`**

6. **Static methods and pass-by-value** (9 points). Consider the following Java program.

```
public class Greetings
{
   public static void sayHello()
    {
       System.out.print("Hello ");
    }

   public static void sayHello(int x)
    {
       if (x < 10)
          System.out.print("G'day ");
       else
          System.out.print("Howdy ");

       x = x - 5;
    }

    public static void main(String [] args)
    {
       int x = Integer.parseInt(args[0]);
       sayHello(x);
       if (x < 3)
          System.out.println("mate!");
       else
          System.out.println("partner!");
    }
}
```

What will be printed by this program in the following cases?

a. % java Greetings 10

Howdy partner!

b. % java Greetings 7

G'day partner!

c. % java Greetings 2

G'day mate!

7. **Loops and conditionals** (15 points). The following (incomplete) program is supposed to print out an N x M grid of symbols. Each line in the output should contain N symbols, the first and last symbols on each line should be @ and all other symbols should be #. There should be M lines of output. The program reads N from the first command line argument and M from the second command line argument. Assume N ≥ 1 and M ≥ 1. Here are some example runs:

```
% java Grid 6 3
@#####@
@#####@
@#####@
```

```
% java Grid 3 3
@#@
@#@
@#@
```

```
% java Grid 2 2
@@
@@
```

```
% java Grid 1 1
@
```

Complete the missing output section of the program.

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

      for (int i = 0; i < M; i++)
      {
         for (int j = 0; j < N; j++)
         {
            if ((j == 0) || (j == (N - 1)))
               System.out.print("@");
            else
               System.out.print("#");
         }
         System.out.println();
      }

   }
}
```

8. **Java expressions** (6 points). The volume of a ponderosa pine in western Montana is given by the equation:

$V = (10^{-2.3582}) \cdot (D^{1.87254}) \cdot (H^{0.89306})$

where `V` is the volume of the tree in cubic feet, `D` is the tree diameter in inches, and `H` is the height in feet.

Assume your Java program has the tree diameter stored in a variable named `D` and the height in a variable named `H`. Recall that the Java Math class has a static method: `double pow(double a, double b)` which returns `a` to the power of `b`. Show how to declare and initialize a variable to contain the tree's volume.

```
double V = Math.pow(10.0, -2.3582) *
           Math.pow(D, 1.87254) *
           Math.pow(H, 0.89306);
```

9. **Conditionals** (9 points). The partially completed program below is supposed to print out whether a number is to the left, to the right, or inside a number range. The program takes three integer command line arguments, N (the number in question), L (the left bound of the number range), and R (the right bound of the range). The number range is inclusive of the left and right endpoints (e.g. 3, 4 and 6 are all inside the range where L=3 and R=6). Assume R ≥ L.

   Complete the three missing statements which should appear immediately before the three curly brace blocks containing the output statements. For full credit, **use as few comparisons as possible**.

```
public class NumberRange
{
   public static void main(String [] args)
   {
      int N = Integer.parseInt(args[0]);
      int L = Integer.parseInt(args[1]);
      int R = Integer.parseInt(args[2]);

      if (N < L)
      {
         System.out.println(N + " is left of [" + L + ", " + R + "]");
      }

      else if (N <= R)
      {
         System.out.println(N + " is inside of [" + L + ", " + R + "]");
      }

      else
      {
         System.out.println(N + " is right of [" + L + ", " + R + "]");
      }
   }
}
```

Example runs:

```
% java NumberRange 3 5 7
3 is left of [5, 7]

% java NumberRange 10 5 7
10 is right of [5, 7]

% java NumberRange 6 5 7
6 is inside of [5, 7]

% java NumberRange 7 5 7
7 is inside of [5, 7]
```

10. **Input and using objects** (12 points). Here is the API for a class that counts the frequency of words:

```
public class WordFreq
-------------------------------------------------------------------------
       WordFreq()                    // create a blank object with no word counts
void   addWord(String str)           // add one to the count for the word str
void   addWord(String str, int num)  // add num to the count for the word str
int    getCount(String str)          // returns # of times the word str was found
```

You need to create a client program that reads a file from standard input and counts all the words occurring in the file. It then prints out the count of the 0 or more words specified as command-line arguments. Write the letter of the code fragment that needs to go in each box to produce a correct program. Not all code fragments will be used. An example text file and execution is shown on the right.

```
public class WordFreqClient
d
   public static void main(String [] args)
   {
      // Create an object to track the data
      [ F ]
      // Read in all the data from standard input
      [ A ]
      {
         // Get the next word
         [ D ]
         // Update the count for this word
         [ H ]
      }

      // Loop over all words from the command-line and print each word's count
      [ N ]
      {
         [ K ]
      }
   }
}
```

Example run:

```
% more freq.txt
the cat sat on the mat
the bat is not on the mat

% java WordFreqClient the cat
mat < freq.txt
the 4
cat 1
mat 2
```

A. `while (!StdIn.isEmpty())`

B. `while (StdIn.readString())`

C. `String word = args[i]`

D. `String word = StdIn.readString();`

E. `WordFreq freq = new WordFreq(args[0]);`

F. `WordFreq freq = new WordFreq();`

G. `freq.addWord(word, i);`

H. `freq.addWord(word);`

I. `addWord(word);`

J. `freq++;`

K. `System.out.println(arg + " " + freq.getCount(arg));`

L. `System.out.println(args[i] + " " + freq.getCount(args[i]));`

M. `while (String arg : args)`

N. `for (String arg : args)`

O. `for (int i = 0; i <= args.length; i++)`

11. **Creating objects** (15 points). You are tasked with completing a class that represents a balloon. A balloon has a current volume of air, a maximum volume of air, and some text that appears on the balloon (e.g. "Happy Birthday!", "It's a girl!", etc). The volume of air is a floating-point value. The current class is missing its instance variables and the implementation of its methods. Fill in the missing parts.

```
public class Balloon
{
   private static final double VOL_PER_PUMP = 1.3;  // Volume of air per pump

   // Instance variables
   private double currentVol = 0.0;
   private double maxVol     = 0.0;
   private String label      = "";









   // Create a new empty balloon with the given maximum volume and text
   public Balloon(double maxVolume, String text)
   {
      this.maxVol = maxVolume;
      this.label  = text;









   }

   // Getter method for the current volume of the balloon
   public double getCurrentVolume()
   {
       return currentVol;




   }
```

(continued on the next page)

11. **Creating objects (continued)**

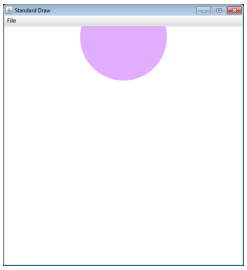
```
    // Pump up the balloon a certain number of times.
    //
    // A single pump adds a volume of air specified by the constant VOL_PER_PUMP.
    // If the pumping results in exceeding the maximum volume, the balloon
    // pops and the current and maximum volume are set to 0.
    //
    // No pumping should occur if numPumps is <= 0.
    //
    // Returns the current volume after the pumping is completed.
    public double pump(int numPumps)
    {
       if (numPumps <= 0)
          return currentVol;
       currentVol += (VOL_PER_PUMP * numPumps);
       if (currentVol > maxVol)
       {
          maxVol = 0.0;
          currentVol = 0.0;
       }
       return currentVol;

    }
}
```

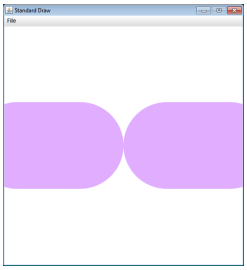
12. Objects and drawing (10 points). Here is a subset of the API for the `Ball` class we saw in lecture:

```
public class Ball
---------------------------------------------------------------------------
       Ball(double x, double y, double r)     // create a ball at (x,y) with radius r
  void draw()                                 // draw the ball
boolean overlap(Ball other)                   // does ball intersect with another?
  void move(double deltaX, double deltaY)     // move the ball by deltaX and deltaY
```

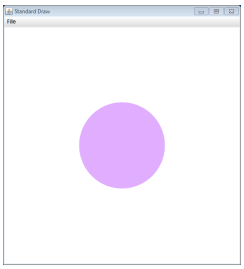
On the left are the final outputs from a number of programs. Label each output with the letter of the program that generated it (each letter is used exactly once). All programs used the default `StdDraw` coordinates with (0.0, 0.0) being in the lower-left corner and (1.0, 1.0) being in the upper-right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



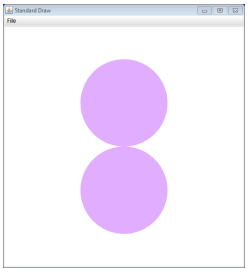
__B__
Program



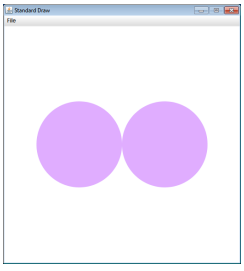
__A__
Program



__E__
Program



__D__
Program



__C__
Program

Program A

```
Ball b1 = new Ball(1.0, 0.5, 0.2);
Ball b2 = new Ball(0.0, 0.5, 0.2);
while (!b1.overlap(b2))
{
   b1.move(-0.01, 0.0);
   b2.move(0.01, 0.0);
   b1.draw();
   b2.draw();
}
```

Program B

```
Ball b1;
Ball b2 = new Ball(0.5, 1.0, 0.2);
b1 = b2;
b2 = new Ball(0.5, 0.0, 0.2);
b1.draw();
```

Program C

```
Ball b1 = new Ball(1.0, 0.5, 0.2);
Ball b2 = new Ball(0.0, 0.5, 0.2);
while (!b1.overlap(b2))
{
   b1.move(-0.01, 0.0);
   b2.move(0.01, 0.0);
}
b1.draw();
b2.draw();
```

Program D

```
Ball b1 = new Ball(0.5, 0.5, 0.2);
Ball b2 = new Ball(0.5, 0.5, 0.2);
b1.move(0.0, -0.2);
b2.move(0.0, 0.2);
b1.draw();
b2.draw();
```

Program E

```
Ball b1 = new Ball(0.5, 0.5, 0.2);
Ball b2 = new Ball(0.5, 0.5, 0.2);
b2.move(0.0, 0.2);
b1 = b2;
b1.move(0.0, -0.2);
b1.draw();
b2.draw();
```