CS201 **Name: _____**
Midterm Exam, Total = 100 Points

Please write neatly and show as much of your work as possible for partial credit. Scan through all problems first, and attack the easiest problems first. Use the last page or the back if you need more space.

1. (15 pts) Short Answer. Provide brief (1-3 sentence) answers to the following:

a) What is the role of a stub in modular program design?

b) If s1 and s2 are both String variables, why doesn't s1==s2 compare if the contents of the two strings are identical?

c) What is the difference between an interpreted programming language and a compiled programming language?

d) If a Java program is created and compiled on a Windows machine, is it possible to run the resulting class file on a Macintosh or Unix machine? Explain your answer.

e) How do you compile and run a Java program named "foo.java" under Unix?

2. (6 pts) Operator Precedence.

Show the output of the following code fragment:

```
System.out.println(2 + 2 % 2 - 2 * 2 / 2);
System.out.println(1 / 4 + 1.0);
System.out.println(2.0 / 4 + (int) 1.8);
```

3. (12 pts) Find the Bugs

All of the following code snippets have a bug. Identify each one and fix the bug. Assume the necessary code is in place to make a working program (e.g. a main method, a class, etc.).

a) The following code should loop indefinitely until the user types in "yes":

```
BufferedReader inFromUser = new BufferedReader(
                  new InputStreamReader(System.in));
String s;
do {
   System.out.println("Give up?");
   s = inFromUser.readLine();
} while (s != "yes");
```

b) The following should continue until the user enters a number from 1-3:

```
BufferedReader inFromUser = new BufferedReader(
                  new InputStreamReader(System.in));
String s;
int i = 0;
while ((i!=1) || (i!=2) || (i!=3))
{
   System.out.println("Enter either 1, 2, or 3");
   s = inFromUser.readLine();
   i = Integer.parseInt(s);
}
```

c) The following should sum the numbers from 1 to 10:

```
int sum=0, ctr=1;
while (ctr <= 10)
   sum = sum + ctr;
   ctr = ctr + 1;

}
```

d) The following code in main should invoke the method named "Foo":

```
   class Stuff {
         public static void main(String[] args) {
               Stuff.Foo();
         }
         public void Foo() {
               System.out.println("Bah.");
         }
   }
```

4. (4 pts) Short Circuit Evaluation.

For the following if-then statement, cross out the expressions that are **not evaluated, if any,** due to the short circuit behavior of Java.

```
int x=3,y=2,z=1;
if ((y<3) && ( x-y <1) && (5/(z-1)<0) && (y+z >0))
{
   System.out.println("Mmm, pizza");
}
```

5. (20 pts) Loops.

a) The following code processes 10 numbers:

```
int sum, i;
for (i=1, sum=0; i<=10; i++)
{
   sum = sum + i*i*i;
   System.out.println(sum);
}
```

Rewrite the code above such that the functionality remains the same, but use a while loop instead of a for loop.

b) Write a nested loop that outputs a right triangle of *'s with sides of length n, where n is a variable greater than or equal to 1. For example, if n = 3, then your code should output:

```
*
**
***
```

If n=4 then your code should output:

```
*
**
***
****
```

Write the code that would go inside a main() method to accomplish the above. Assume that the value *n* is hardcoded in the program (e.g., at the top of the program is the declaration: int n=4; if we wanted a triangle with sides of length 4).

6. (14 pts) Writing Methods, Passing Parameters.

Write a method called **FindDistance** that takes as input four *integers*, x1, y1, x2, y2, where (x1,y1) and (x2,y2) specify points on a two-dimensional grid. Your method should return as a *double* the distance between the two points using the formula:

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

For example, when your method is added to the class with the main method below, the output should be:

5.0
7.07

```
class Test {
      // The FindDistance method would be inserted here

      // Sample main method
      public static void main(String[] args)
      {
            Test myObj = new Test();
            // x1 = 0, y1=0, x2=3, y2=4
            System.out.println(myObj.FindDistance(0,0,3,4);
            // x1 = -2, y1 =0, x2=3, y2=5
            System.out.println(myObj.FindDistance(-2,0,3,5);
      }
}
```

7. (14 pts) Scoping and Parameters.

Give the output of the following program when it is compiled and executed. Show your work wherever possible in tracing through the values assigned to all variables for partial credit in the event your final answer is incorrect. Don't forget to that the code in the default constructor is executed when a new Class is made with no parameters!

```
class MyNumberClass {
      private int m_num;

      public MyNumberClass() {
            m_num = 10;
      }
      public MyNumberClass(int n) {
            m_num = n;
      }
      public void SetNumber(int n) {
            m_num = n;
      }
      public int GetNumber() {
            return m_num;
      }
}

class CS201Test {
  public static void main(String[] args) {
            MyNumberClass n1 = new MyNumberClass();
            MyNumberClass n2 = new MyNumberClass(5);
            int n3=0;

            System.out.println(n1.GetNumber());
            System.out.println(n2.GetNumber() + " " + n3);
            CS201Test.DoIt(n1, n2, n3);
            System.out.println("In main:");
            System.out.println(n1.GetNumber());
            System.out.println(n2.GetNumber() + " " + n3);
      }

  public static void DoIt(MyNumberClass x1, MyNumberClass x2, int n3) {
            int temp = x1.GetNumber();
            n3 = 3;
            x1.SetNumber(x2.GetNumber());
            x2.SetNumber(temp);
            System.out.println("In DoIt");
            System.out.println(x1.GetNumber());
            System.out.println(x2.GetNumber() + " " + n3);
      }
}
```

8. (15 pts) Leap Years

A leap year has only 364 days instead of 364 days. Every year divisible by four is a leap year, with the exception of years divisible by 100 and not divisible by 400.

For example, the following are leap years:

2004 - Divisible by 4. Not divisible by 100, so the exception does not apply.
1600 - Divisible by 4. Divisible by 100 but also divisible by 400 so the exception does not apply.
2000 - Divisible by 4. Divisible by 100 but also divisible by 400 so the exception does not apply.

The following are not leap years:

2003 - Not divisible by 4.
1700 - Divisible by 4. Divisible by 100 and also not divisible by 400 so the exception applies.
1800 - Divisible by 4. Divisible by 100 and also not divisible by 400 so the exception applies.

a) Write a static method that takes as input an integer representing a year, and returns a boolean indicating if the year is a leap year or not.

b) Write a main method that uses your static method from part a to print a list of all leap years between year 1 and year 3000.

(this page intentionally left blank if you need more space for your answers)