Recall the Academic Integrity statement that you signed. Write all answers clearly on these pages, ensuring your final answers are easily recognizable. The number of points for each problem is clearly marked, for a total of 25 points. I will post my solutions on the web on Monday, off the Solutions link, after class.

1. (2.5 pts) Name the 9 kinds of statements that we have learned. Write a * after each control structure.
   (a)
   (b)
   (c)
   (d)
   (e)
   (f)
   (g)
   (h)
   (i)

2. (2.5 pts) Assume that we declare `int x=1,y=2;` each time before Java executes each of the following statements; fill in the states for these variables after each statement executes (or write NLES for not legal expression statement).
   
   \[
   \begin{align*}
   &y == x; \\
   &x = y += 2; \quad &y = (2*x)++; \quad &x = x++; \quad &y = 1 + (x = ++ y);
   \end{align*}
   \]

   x is now y is now x is now x is now x is now
   y is now y is now y is now y is now y is now

3. (2 pts) Given the declaration `String s;` and the pair of prototypes `void System.out.print(String)` and `String Prompt.forString(String)`, mark each expression statement legal or illegal (if illegal, explain why).
   (a) `s = Prompt.forString("Enter s");`
   (b) `Prompt.forString("Enter s");`
   (c) `s = System.out.print("Enter s");`

4. (5 pts) Write block whose statements, when executed, print the numbers from 1 to 100 (separated by one space), starting a new line after printing every 10th value (including the last; see box below). For full credit, write the simplest code (try to use just one general `for` loop). Hint: use `print` and `println` and control structures to solve a smaller problem: write a block that prints the numbers 1-15, printing 5 per line and hand simulate your code to ensure it executes correctly; then upgrade it to print the numbers 1 to 100 printing 10 per line.

5. (1 pt) Would you rather have a syntax error or execution error in your program (briefly explain why)?
6a. (5 pts) Draw a box around **every Java statement** in the code that appears below. Do this neatly.

6b. (5 pts) Hand simulate the code that appears below (the boxing from part 6a might help you) by filling in the **compact trace table** on its right. Remember, you do not need to indicate which statement changed the state or performed input/output; just use the next available line in each column to show state changes or input/output, as in the compact trace tables that we studied in class. When prompted, the user will enter the values **43** and **2**

Hand simulate this code carefully; one mistake can snowball. Hint: there are extra lines in the compact trace table, so you won’t need to fill in them all.

```java
{ 
    int a = 0, c = 0;
    int x = Prompt.forInt("Enter x");
    int y = Prompt.forInt("Enter y"); m = y;
    for (; ; ) {
        if (x == 0)
            break;
        else {
            if (x%2 == 1) {
                c++;
                a = a + m;
            }
            x = x/2;
            m = m * y;
        }
    }
    System.out.println("a=\"a\"+c=\"c\"");
}
```

7. (2 pts) What does the following code fragment print when it reads a file containing: **1 2 a 4 5 2 a 6 0 0**

Recall that when `.nextInt()` is called on a it throws a `NumberFormatException`; when `.nextInt()` is called and there are no more values in the file, it throws `EOFException`

```java
int x = 0, y = 0;
for (; ; )
    try {
        x = inputFile.readInt(); y = inputFile.readInt(); System.out.print(10*x/y);
    } catch (NumberFormatException nfe ) {System.out.print(":+x+:");}
    catch (EOFException efoe) {System.out.print("E"); break;}
    catch (ArithmeticException eae ) {System.out.print("A");}
System.out.print("D");
```

It prints: