COS 126 **Mini Practice Exam** **30** min

Data Types. Give the value and type of each of the following Java expressions. If an expression will not compile or will cause an exception at runtime, put an **X** under value and type. If the value is a string, enclose it in double quotes.

Expression	Value	Type
`1 / 0`		
`"800" * 1`		
`"1" + " - " + "1"`		
`3.14159 + (int) Math.PI`		
`1-1-1-1`		
`3 / 2.0 + 2 * 5`		
`(8 <= 2) || (2e8 <= 8e2)`		
`Double.parseDouble("8.5*2")`		
`"1" + 1 + 1 + "1"`		

Syntax & Scope. Consider the following code.

```
public class Cubes
{
    public static int cube(int i)
    {   return i * i * i;   }

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

Among the following statements, circle those that are true.

A. Will not compile because braces in `cube()` are not on separate lines.

B. Will not compile because braces are missing in the `for` loop.

C. Will not compile because `i` is not declared in `cube()`.

D. Prints only a few lines because `cube()` rapidly increases the `i` used by `main()`.

E. Prints the squares of the integers from 1 to 1000.

F. Prints the cubes of the integers from 1 to 1000.

G. Goes into an infinite loop.

Methods & Arrays. Consider the following code:

```
public class MethodTester {
   private static void methodB(int[] c, int d) {
      c[0]++;
      d += 42;
   }
   private static int methodA(int[] a, int b) {
      methodB(a, b);
      a[0]++;
      return b/2;
   }
   public static void main(String[] args) {
      int[] arr = {8, 9, 10};
      int x = 1;
      x = methodA(arr, x);
      System.out.println(arr[0] + " " + x);
    }
}
```

Which one of the following is the output of this program?

"8 3"

"8 10”

“8 21”

"9 1"

"9 3"

“9 21”

"10 0"

"10 1"

“10 21"

Recursion. Consider the following code:

```
public class Series {
  public static int func(int j) {
  if (j==1) return 1;
    return 2 * func(j - 1) + 5 * func(j - 2);
  }

  public static void main(String[] args) {
    int N = Integer.parseInt(args[0]); // assume N >= 0

    System.out.println(func(N));
  }
}
```

a. Draw the recursion tree for `func(3)`. You only need to draw the tree up to 3 levels.

b. What is the problem with this recursive program, based on the tree you drew above?

Recursive Graphics. Complete the following recursive graphics program using the following lines of code such that the program produces each of the pictures below. For the sake of simplicity, you do not have to worry about setting the pen color.

```
public class Squares {
    public static void recur(int N, double x, double y, double s) {
        if (N == 0) return;

        // What goes here?
    }

    public static void main(String[] args) {
        recur(3, .5, .5, .25);
    }
}
```

A. `recur(N-1, x+s, y,   s/2);` `// Recur right`
B. `recur(N-1, x-s, y,   s/2);` `// Recur left`
C. `recur(N-1, x,   y+s, s/2);` `// Recur top`
D. `recur(N-1, x,   y-s, s/2);` `// Recur bottom`
E. `recur(N-1, x+s, y+s, s/2);` `// Recur top right`
F. `recur(N-1, x-s, y+s, s/2);` `// Recur top left`
G. `recur(N-1, x+s, y-s, s/2);` `// Recur bottom right`
H. `recur(N-1, x-s, y-s, s/2);` `// Recur bottom left`
I. `StdDraw.filledSquare(x, y, s);` `// Draw square at (x, y)`

In the boxes below, provide **an ordered sequence of 3-4 letters** corresponding to the 3-4 lines of code that you would need to add to the `recur()` method to produce each picture.