COS126 Mini Practice Exam

Q1: 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. If the value is a string, enclose it in double quotes.

Expression	**Value**	**Type**
1 / 0	**X**	
"800" * 1	**X**	
"1" + " - " + "1"	**"1-1"**	**String**
3.14159 + (int) Math.PI	**6.14159**	**double**
1-1-1-1	**-2**	**int**
3 / 2.0 + 2 * 5	**11.5**	**double**
(8 <= 2) || (2e8 <= 8e2)	**false**	**boolean**
Double.parseDouble("8.5*2")	**X**	
"1" + 1 + 1 + "1"	**"1111"**	**String**

Q2: 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? Circle your answer.

8 3

8 10

8 21

9 1

9 3

9 21

[✓ 10 0]

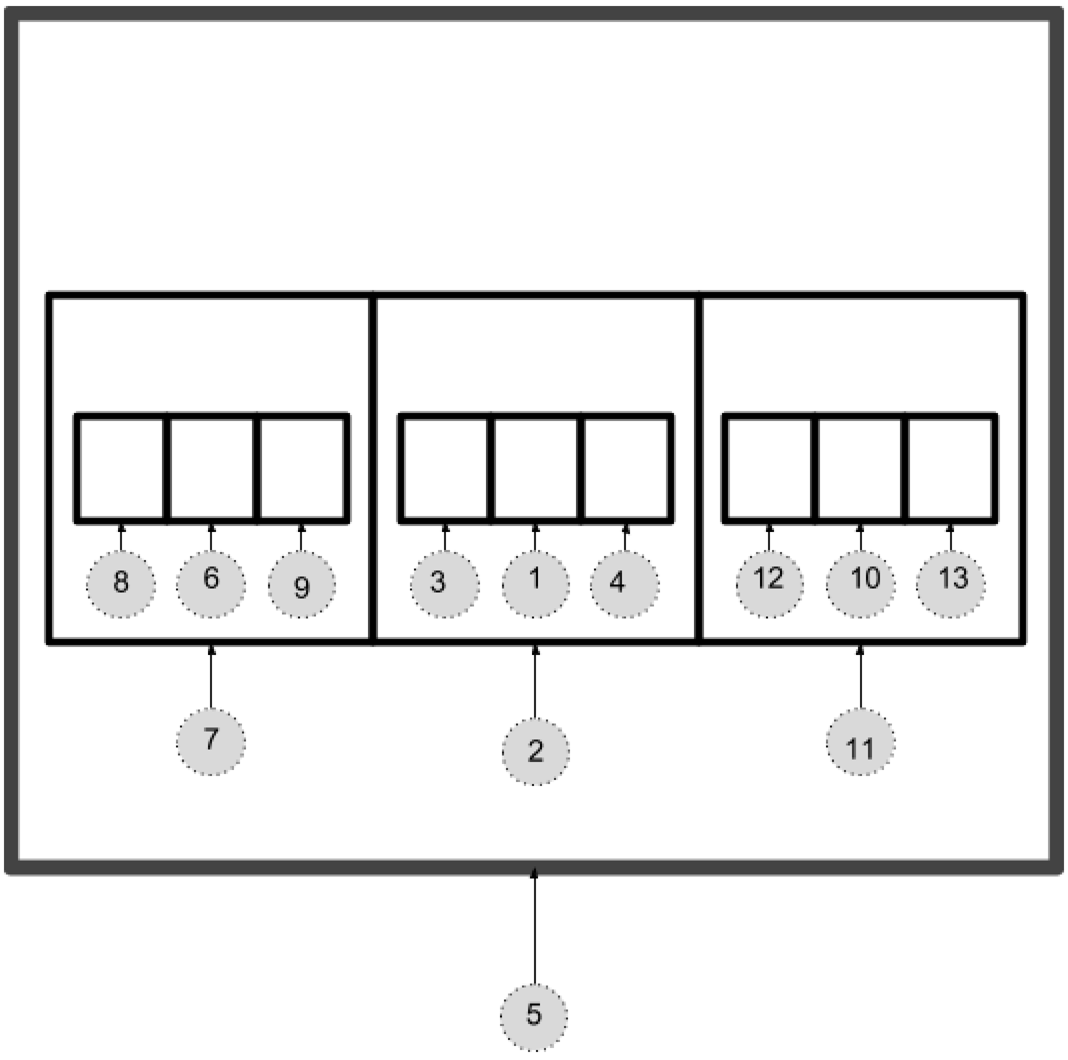
10 1

10 21

Q3: Here is a method that draws squares recursively:

```
public static void draw(int n, double x, double y, double r) {
  if (n==0) return;          // base case
  draw(n-1, x, y, r/4);
  StdDraw.square(x, y, r);   // draw a square
  draw(n-1, x - r/2, y, r/4);
  draw(n-1, x + r/2, y, r/4);
}
```

Below, we plot the picture produced when `draw(3, 0.5, 0.5, 0.5)` is called. It draws thirteen squares, which we have also <u>labelled</u> with dashed circles and arrows. What is the order in which the squares were drawn? Write all of the integers from 1 to 13 in the circles to indicate this order, with 1 labelling the first square drawn and 13 the last.



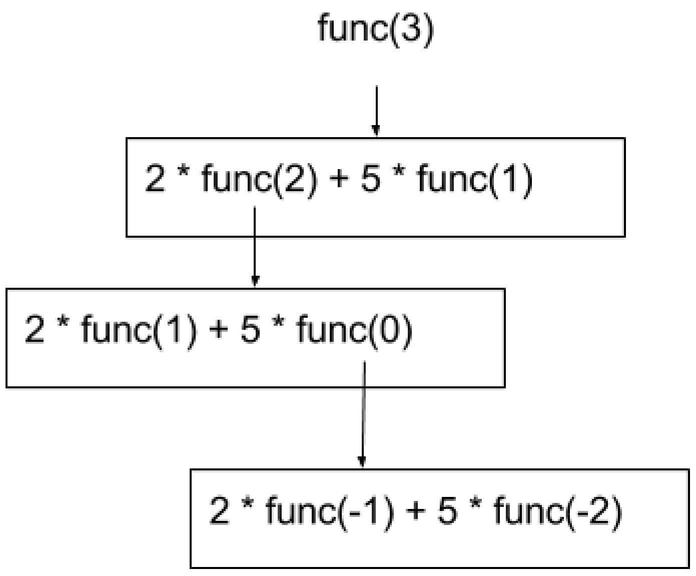
Q4: Recursion: Consider the following program:

```
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, which means the height of the recursion tree should be no greater than 3.**


func(3)
2 * func(2) + 5 * func(1)
2 * func(1) + 5 * func(0)
2 * func(-1) + 5 * func(-2)

b. From the recursion tree in (a), do you see a problem with the program? Explain what is the problem.

The problem is the the function reductive step skips over the base case which will result in a stack overflow error.

```
Change if (j == 1) return 1; to if (j <= 1) return 1;
```