APCS – Notes on Recursion

A recursive method is a method that calls itself. An iterative method is a method that uses a loop to repeat an action. Recursion is just another way to repeat an action.

Remember that:

- ***Iteration*** **employs loop structures, such as for, while and dowhile structures, to perform a repetitive algorithm.** ***Recursion*** **employs a recursive method to perform repetition, where a recursive method is one that calls itself.**

- **A** ***recursive method*** **is a method that calls or invokes itself, usually with a different argument, as a means of performing some repetitive algorithm. A** ***recursive definition*** **is one in which the nth term in a sequence is defined in terms of the n-1st term.**

- **A** ***base case*** **is that part of a recursive definition or method that does not contain a recursive call. It serves to limit or bound the process of repetition. A** ***recursive case*** **is that part of a recursive definition or method that does involve a recursive call.**

- **Recursive algorithms use method calls to effect repetition. A loop, such as a for loop, does not. Method calls involve more computational overhead than a for loop, because a block representing the method call must be placed on the method call stack. This causes recursive algorithms to be somewhat less efficient than iterative ones.**

Determine the output of the following programs.

1. **Write a recursive method that takes a single positive `int` parameter n and prints the sequence of even numbers between n down to 0.**

```
/**
* Precondition: n > 0
* Postcondition: none
*/
public class PrintEvensDemo
{
    public static void main( String[] args )
    {
        int n = 8;
        printEvens(8);
    }

    public static void printEvens( int k )
    {
        if(k < 2 )
            System.out.println(); // base case
        else
        {
            if( k % 2 == 0 )
            {
                System.out.print(k + " ");
                printEvens(k - 1); // recursive case
            }
            else printEvens(k - 1); // recursive case
        }
    }
}
```

2. **Write a recursive method that takes a single int parameter a and prints the multiples of n between 0 and a.**

```
public class MultiplesDemo
{
    public static void main( String[] args )
    {
        System.out.print("The positive multiples of 3 less than " +
                         " 30 are:\n\n");
        showMultiples(3, 30);
    }
    public static void showMultiples( int n, int a )
    {
        if( a < n )
        {
            System.out.println("\n\n\tDONE");
        }
        else
        {
            if( a%n == 0 ) System.out.print("\t" + a);
            showMultiples( n, a-1 );
        }
    }
}
```

3. **Write a recursive method to print the following geometric pattern:**

```
#
##
###
####
#####
```

```
public class Pattern1
{
    public static void main( String[] args )
    {
        Pattern1 pretty = new Pattern1();
        int n = 5;
        pretty.showPattern(n);
        System.out.print("\nDONE");
    }

    public void showPattern( int k )
    {
        if( k <= 0 )
        {
            return;
        }
        else
        {
            showPattern( k - 1 );
            for( int j=0; j<k; j++) System.out.print("# ");
            System.out.println();
        }
    }
}
```

4. **What is the output of the following program?**

```
public class Mystery1
{
    public static void main( String[] args )
    {
        int d = 29417;
        int od = d;
        int n = 0;
        Mystery1 unknown = new Mystery1();
        unknown.someMystery1(d, n);
    }

    public void someMystery1( int h, int r )
    {
        if( h <= 0 )
        {
            r += h;
            System.out.print("r = " + r );
        }
        else
        {
            r += h%10;
            someMystery1( h/10, r );
        }
    }
}
```

5. **What is the output of the following program?**

```
public class Mystery3
{
    public static void main( String[] args )
    {
        int n = 5872;
        System.out.println("something = " + something(n));
    }

    public static int something( int k )
    {
        if( k < 10 )
        {
            return k;
        }
        else
        {
            return k%10 + something( k / 10 );
        }
    }
}
```

6. **What is the output of the following program?**

```
public class Mystery2
{
    public static void main( String[] args )
    {
        someMystery2(3, 30);
    }

    public static void someMystery2(int n, int b)
   {
      if( n > b )
      {
         System.out.print("\n\nEnd of recursion\n\n");
      }
      else
      {
         someMystery2( n+4, b-3 );
         if( b%n != 0 )
         {
            for(int k = 0; k<(b/n); k++) System.out.print("* ");
            System.out.println();
          }
      }
   }
}
```

7. **What is the output of the following program?**

```
public class Mystery4
{
    public static void main( String[] args )
    {
        String name = "boom";
        System.out.print(findValue(name));
    }

    public static int findValue( String k )
    {
        if( k.length()==0 )
        {
            return 0;
        }
        else
        {
            return k.charAt(0) + findValue( k.substring(1) );
        }
    }
}
```