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 do-while 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 \texttt{int} parameter \texttt{n} and prints the sequence of even numbers between \texttt{n} down to 0.

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

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

```java
public class MultiplesDemo {
    public static void main( String[] args ) {
        System.out.print("The positive multiples of 3 less than " +
            " 30 are:
        ");
        showMultiples(3, 30);
    }
    public static void showMultiples( int n, int a ) {
        if( a < n ) {
            System.out.println("DONE");
        } 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:

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

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

```java
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?

```java
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?

```java
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?

```java
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) );
        }
    }
}
```
ANSWERS

1. 8 6 4 2

2. The positive multiples of 3 less than 30 are:
   30 27 24 21 18 15 12 9 6 3
   DONE

3. 
   #
   ##
   ###
   ####
   #####
   ######
   DONE

4. r = 23

5. something = 22

6. End of recursion
   *
   * *
   * *
   * * *

7. 429