Recursion Summary

Recursion is a method calling on itself to solve a simpler problem. With every recursive call the problem is simpler than the original problem. Recursion involves internal use of stacks. When a recursive call is made the current computation is pushed in a stack. When the base case is reached the computation that is on the top of the stack is popped and executed. This continues until the stack has no more computations left. Therefore, the computation below is read from right to left by the computer:

*fact (4) = 4 * fact (3) = 3 * fact (2) = 2 * fact (1) = 1*

*24 ← 4 * 6 ← 3 * 2 ← 2 * 1*

The base case is the fundamental condition where no further problem solving is necessary:

```
public int fact(int n){
if(n<=1)
        return 1;  //this is the base case.
else
return  n*fact(n-1);  //this is where the recursion occurs.
}
```

Recursion is a good problem solving approach when the problem is difficult to solve using iterations, and recursive algorithms are easy to understand and implement. Recursion is also used to solve a problem by backtracking.

The major pitfalls of using recursion:

1. If there is no base case for the recursive method, then the recursion will go on infinitely until the computer runs out of memory and the program crashes.
2. When programming recursively, you need to make sure that the program is progressing towards the base case, that is, it is getting simpler after every recursion.
3. Recursion generates results slower or at the same speed when compared to iterations for solving simple problems.

Whenever a set of cooperating methods call each other in a recursive fashion it is called mutual recursion. Mutual recursion is highly advanced than the simple recursion.

To use recursion, divide your problem into two parts:

a. The simplest possible case you can answer (base case).
b. All the other complex cases that progress towards the base case with each recursive call.