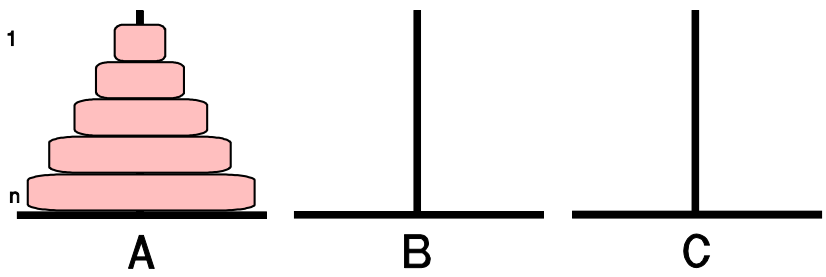
Towers of Hanoi

Move all the disks from peg A to peg C.

1. Can only move one disk at a time
2. Can only place a smaller disk on top of a larger one



Solution:

1. Move top n-1 disks to peg B
2. Move disk n to peg C
3. Move n-1 disks on peg B to peg C

RECUR005

University of Idaho

Towers of Hanoi – Computer Solution

```
const int N = 64; // number of disks (don't really try 64!)

int main()
  {
    toh(N, 'A', 'C', 'B');
  }  // END main


void toh(int count, char start, char finish, char temp)
  {
   if(count > 0)
     {
      toh(count-1, start, temp, finish);
      cout << "Move disk " << count << "from peg " << start
           << " to peg " << finish << endl;
      toh(count-1, temp, finish, start);
     }  // END if
  }  // END toh
```

RECUR007

University of Idaho

Mathematical Definition of Factorial

$$n! = \begin{cases} 1 & \text{, for } n = 0 \\ n * (n-1)! & \text{, for } n > 0 \end{cases}$$

This is called a "recurrence relation."

Necessary Characteristics for Using Recursion

1. Problem solution can be expressed as a "smaller" version of itself.

2. Stopping case - At least one (often trivial) case can be computed non-recursively.

C++ Implementation of Recursion

1. *When a subroutine is invoked, storage for arguments and local variables is allocated in a memory area called an "activation record"*

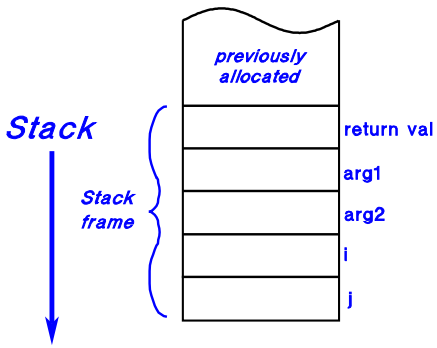
2. *Activation records are stored on the "stack."*

3. *Upon return, activation record is deallocated.*

4. *Activation record is also called a "stack frame"*

Example of an Activation Record

```
int func1 (int arg1, int arg2)
  {
    int i, j;
      .
      .
      .
    return(j);
```



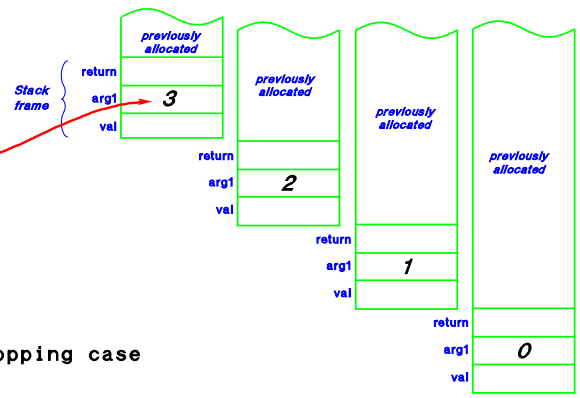
Recursion Example

```
void main()
  {
   int i = 3, j;
    .
    .
   j = fact(i);
    .
  } //END main

int fact (int arg1)
  {
   int val;

   if(arg1 <= 0) //stopping case
     val = 1;
   else
     val = arg1 * fact(arg1-1); //recursive call

  return val;
 } // end fact
```



RECUR050