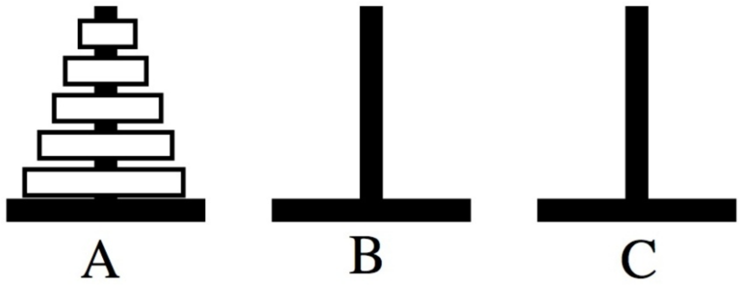
Recursive Procedures in Computer Programming

In the "Towers of Hanoi" problem, we have three pegs, one of which has a stack of disks on it, with the size of the disks increasing from top to bottom, i.e. the smallest disk is on top. We want to move that stack of disks to some other peg. We are allowed to move only one disk at a time, and we can never place a larger disk on top of a smaller one.



Let's call the three pegs A, B, and C. Suppose we have a stack of N disks on peg A, and we want to move the stack to peg C. A general strategy to move the stack of N disks could be:

1. Move a stack consisting of the top (N – 1) disks from peg A to peg B
2. Move the one remaining disk on peg A to peg C
3. Move the stack of (N – 1) disks from peg B to peg C.

The computer instruction to move the stack from A to C could look something like this:

```
MoveStack(A, C, N)
```

This code assumes that there is a procedure called "MoveStack" which knows how to move a stack (or just the top part of the stack) consisting of a certain number of disks from one peg to another.

The interesting part of the program is the "MoveStack" procedure:

```
PROCEDURE MoveStack(FromPeg, ToPeg, N)
    IF N = 1 THEN
       MoveDisk(FromPeg, ToPeg)
    ELSE
       OtherPeg = (the peg that is neither FromPeg nor ToPeg)
       MoveStack(FromPeg, OtherPeg, N - 1)
       MoveDisk(FromPeg, ToPeg)
       MoveStack(OtherPeg, ToPeg, N - 1)
END
```

Notice that the code for the "MoveStack" procedure contains two calls to that *same* procedure. A procedure which contains calls to itself is called a *recursive procedure*. The behaviour of the procedure is defined partly in terms of itself.