Note on Towers of Hanoi

The Towers of Hanoi is a children's game, played with three pegs and a number of different-sized disks. Let us suppose, S, A, and D be 3 pegs. Also let there are finite number n of disks with decreasing size on peg S. This is pictured in the figure shown below for the case n = 3.

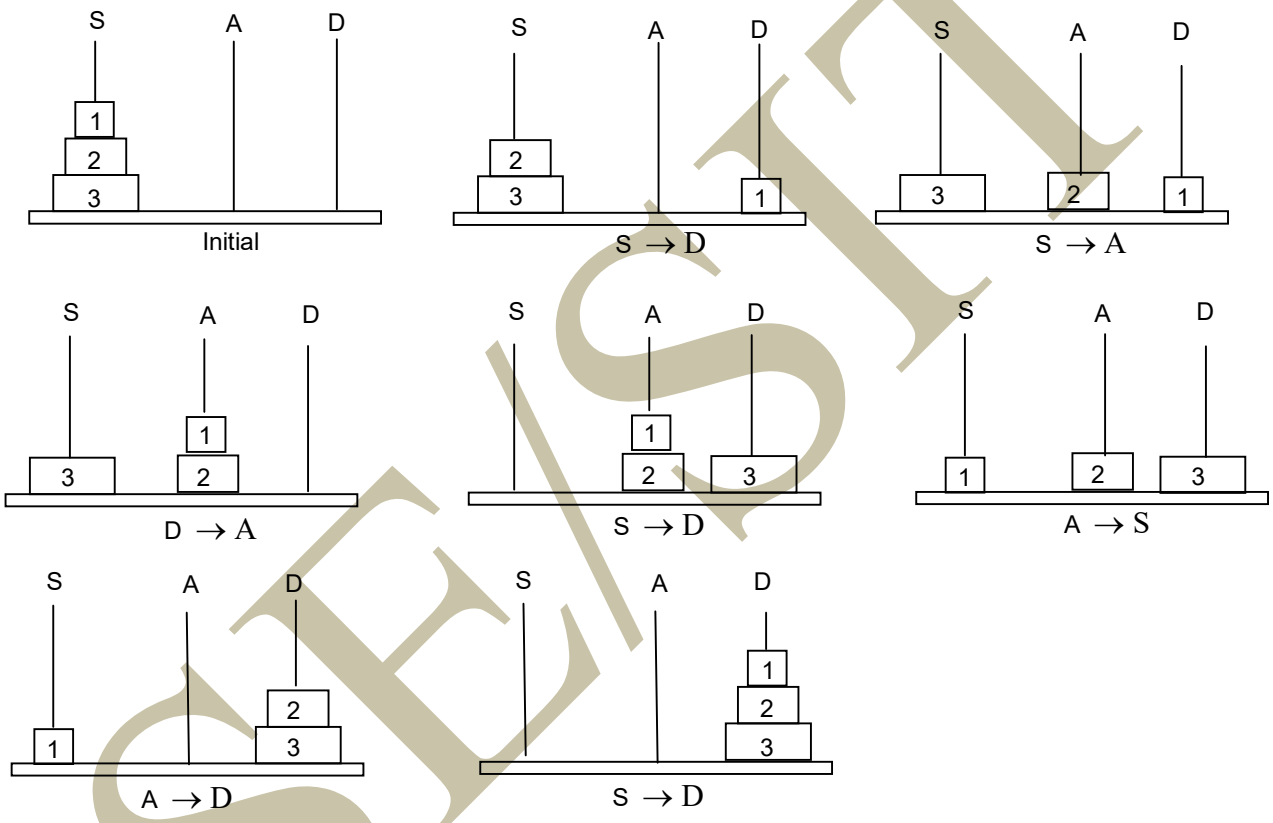
Objective of the game: Move all the disks from peg S to peg D using peg A as an intermediate peg.

Rules are –

1. Only the top disk on any peg may be moved to any other peg.
2. It is not allowed to place a larger disk on a smaller disk.

Representation –

S → D means move **top disk** from **peg S** to **peg D**.



So moves are S → D, S → A, D → A, S → D, A → S, A → D, S → D.
Number of moves = $7 = 2^3 - 1$.
Therefore, for n disks, number of moves = $2^n - 1$.

```
The solutions to Towers of Hanoi problem for n = 1 and n = 2 are -
  For n = 1: Move disk 1 from peg S to peg D.
  For n = 2: Move disk 1 from peg S to peg A.
             Move disk 2 from peg S to peg D.
             Move disk 1 from peg A to peg D.

Thus the problem of moving n disks from peg S to peg D can be specified in the
following recursive manner -
   1. Move top n-1 disks from peg S to peg A.
   2. Move the nth  disk  (the largest disk)from peg S to the peg D
   3. Move the n-1 disks from peg A to peg D.
```

Note on Towers of Hanoi

Recursive definition:

$$\text{Tower}(n, S, A, D) = \begin{cases} \left.\begin{array}{l}\text{Tower}(n-1, S, D, A) \text{ and}\\ \text{Move } n^{th} \text{ disk from peg S to peg D and}\\ \text{Tower}(n-1, A, S, D)\end{array}\right\} & \text{if } n > 1 \\ \text{Move the disk from peg S to peg D} & \text{if } n = 1 \end{cases}$$

Where Tower (n, S, A, D) denotes "n" disks are moved from peg S to peg D using peg A.

Question: WACP for implementing Towers of Hanoi using recursive function.

```
#include <stdio.h>
int main()
{
  int n;                                    // Variable declaration
  void tower(int, char, char, char);       //  function declaration

  printf("\nHow many disks ? ");
  scanf("%d",&n);

  if(n>0)
      tower(n,'S','A','D');              //function call
  else
      printf("\n Do not waste time, Press any key to exit");
  return(0);
}

void tower(int n, char beg, char aux, char end)
{
   if(n==1)
   {
     printf("\nMove Disk %d from peg %c to peg %c\n", n, beg, end);
     return;
   }
   tower(n-1,beg,end,aux);
   printf("\nMove Disk %d from peg %c to peg %c\n", n, beg, end);
   tower(n-1,aux,beg,end);
}
```

Question : Using **Ackermann function** find A(1,2).

Definition of Ackermann function –

a. if m = 0 then A(m, n) = n + 1.
b. if m ≠ 0 but n = 0 then A (m, n) = A(m-1, 1).
c. if m ≠ 0 but n ≠ 0 then A (m, n) = A(m-1, A(m, n - 1)).

Answer: We have the following 11 steps

1. A(1,2) = A(0, A(1,1))
2. A(1,1) = A(0,A(1,0))
3. A(1,0) = A(0,1)
4. A(0,1) = 1 + 1 = 2
5. A(1,0) = 2
6. A(1,1) = A(0,2)
7. A(0,2) = 2 + 1 = 3
8. A(1,1) = 3
9. A(1,2) = A(0,3)
10. A(0,3) = 3 + 1 = 4
11. A(1,2) = 4