UNIT-4 RECURSION

➢ The function is called recursive function if it calls to itself and recursion is a process by which a function calls itself repeatedly until some specified condition will be satisfied.

➢ A function will be recursive, if it contains following features:
- The function should call itself.
- The function should have stopping condition.

➢ The concept of using recursive function is to repeat the execution of statements many times is known as recursion.
➢ There are two types of recursion: Direct and Indirect recursion.

Note: a recursion use stack for its implementation.

Difference between Recursion and Iteration

Recursion	Iteration (loop)
1. Function that contains a call statement to itself.	1. It calls for Repetition of some process until certain criteria is met.
2. It requires Stack implementation.	2. No use of Stack.
3.Slow process	3. Fast process than recursion.
4. Inefficient memory utilization.	4. Efficient memory utilization.
5. Shorter code, so time efficient.	5. Longer code, consume more time.
6. Example: to find factorial of entered number using recursion.	6. Example: to find factorial of entered number using iteration.

Disadvantages of Recursion
1. It consumes more storage space because of stack implementation.
2. The computer may run out of memory if the recursion calls are not checked.
3. It is not more efficient in term of speed.

Application
- Game development
- Tree Traversal
- Expression conversion
- Binary search algorithm
- used in sorting and searching algorithm

Program to find factorial of entered number using recursive function.

```c
#include<stdio.h>
#include<conio.h>
long int factorial(int n);
void main()
{
    clrscr();
    int n;
    long int x;
    printf("Please enter the number\n");
    scanf("%d",& n);
    x=factorial(n);
    printf("The factorial is %ld",x);
    getch();
}
    long int factorial(int n)
    {
        if(n<=1)
            return 1;
        else
            return (n* factorial(n-1));
    }
```

Without using recursion

```c
long int factorial(int n)
{
long int p=1;
for(i=1;i<=n;i++)
p=p*i;
return p;
}
```

Output
Please enter the number: 5
The factorial is 120

//Program to find sum of all natural numbers from 1 to n using recursive function.

```c
#include<stdio.h>
#include<conio.h>
int sum (int);
```

```c
void main()
{
        int n, s;
        printf("Input a number:");
        scanf("%d",&n);
        s=sum(n);
        printf("\n The sum of natural numbers is %d",s);
        getch();
}

int sum (int n)
{
        if(n<=0)
                return 0;
        else
                return(n+sum(n-1));
}
```

Output
Input a number: 10
The sum of natural numbers is 55

//Program to display Fibonacci series up to nth term (0,1,1,2,3,5,8..........)

```c
#include<conio.h>
#include<stdio.h>
void fseries(int);
void main()
{
        int n,f0=0,f1=1;
        clrscr();
        printf("\n Enter the number of terms in fibonacci series:\t");
        scanf("%d",&n);
        if (n >2)
        {
                printf("%d\n%d",f0,f1);
                fseries(n-2);
        }
        else if (n == 2)
                printf("%d\n%d",f0,f1);
        else if (n == 1)
                printf("%d",f0);
        else
                printf("series not possible");
        getch();
}
```
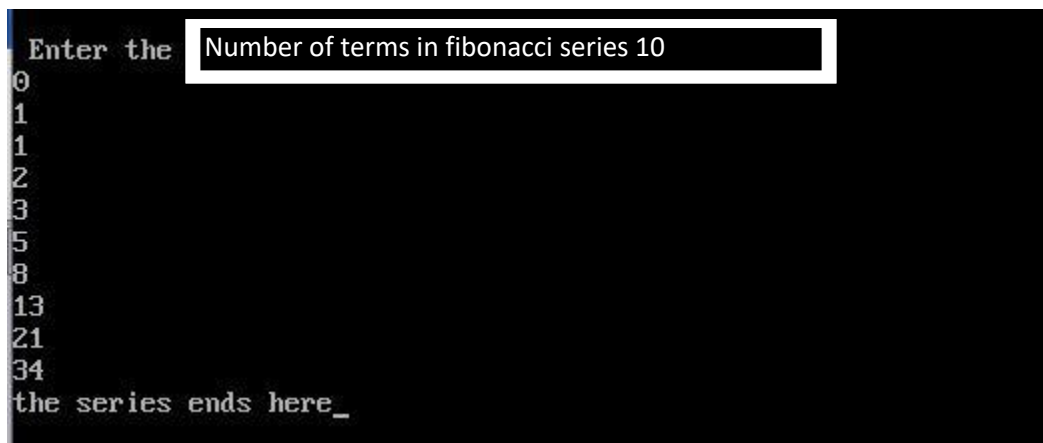
```
void fseries (int p)
{
        int fib;
        static int f0=0, f1=1;
        if (p == 0)
                printf("\nthe series ends here");
        else
        {
                fib = f0 + f1;
                f0 = f1;
                f1 = fib;
                printf("\n%d",fib);
                fseries (p-1);
        }
}
```

Output:

```
Enter the  Number of terms in fibonacci series 10
0
1
1
2
3
5
8
13
21
34
the series ends here_
```

GCD of Two Numbers using Recursion

```c
#include <stdio.h>
int hcf(int n1, int n2);
int main()
{
    int n1, n2;
    printf("Enter two positive integers: ");
    scanf("%d %d", &n1, &n2);
    printf("G.C.D of %d and %d is %d.", n1, n2, hcf(n1, n2));
    return 0;
}

int hcf(int n1, int n2)
{
    if (n2 != 0)
        return hcf(n2, n1 % n2);
    else
        return n1;
}

Enter two positive integers: 366
60
G.C.D of 366 and 60 is 6.
```

The Tower of Hanoi (TOH) and its Solution:

TOH is a famous recursive problem in computer science which is based on three pegs and the set of disk of different size. Each disk has a hole in centre, so that it can be stacked on any peg.

Let A, B, C be the three pegs. At the beginning of the game, the disks are stacked on peg A in such a way that the largest size disk is on the button and smallest sized disk on top.

The problem is to move disk from peg A to peg C, using peg B as auxiliary peg. So, peg A is a source peg & peg C is destination peg.

Rule of Game
1. On transferring the disk from source peg to destination peg at any point, no longer disk can be placed on smaller one.
2. Only one disk can be moved at a time.
3. Each disk must be stacked on only one of pegs.

Example1:

1. Let us take an e.g. where number of disk (n) =2 and show the step by step process.

Note:

Peg A is Source Peg

Peg B is Intermediate Peg

Peg C is Destination Peg
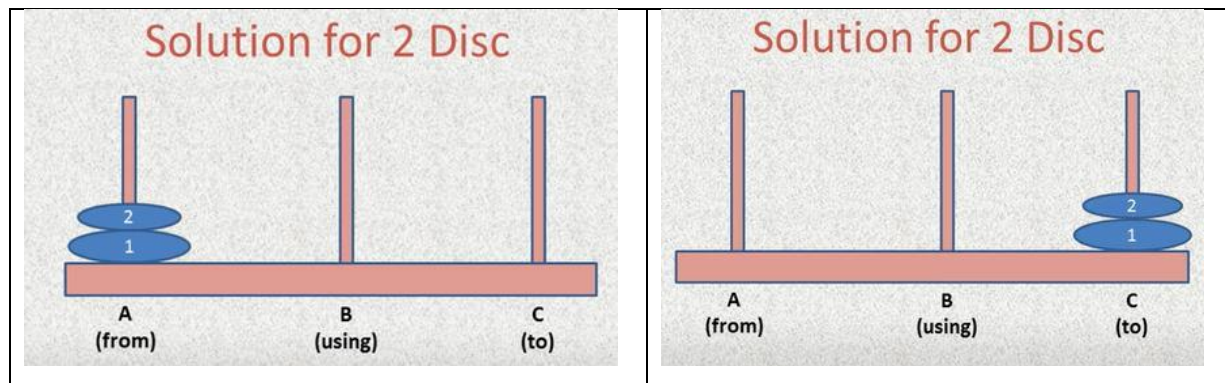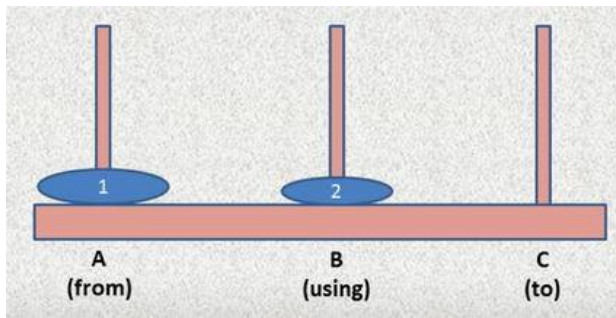

Fig a: Initial condition of disk   **Fig b: Final condition of disk**

So the problem is to move disk1 and disk2 from Peg A to Peg C as shown in the Fig b.
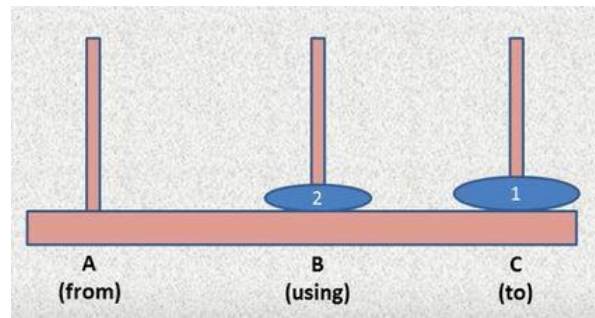
Solution:
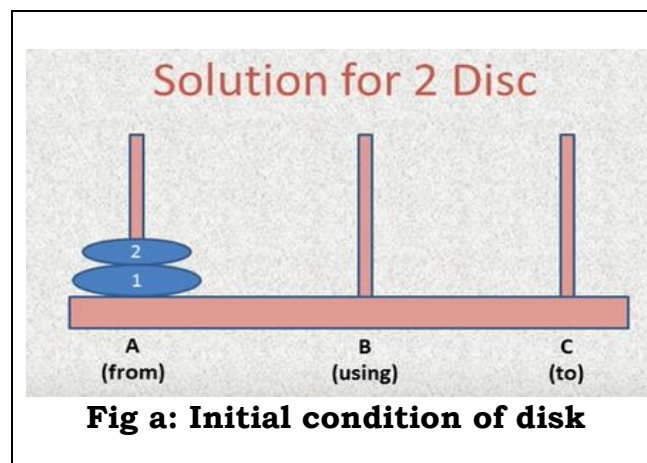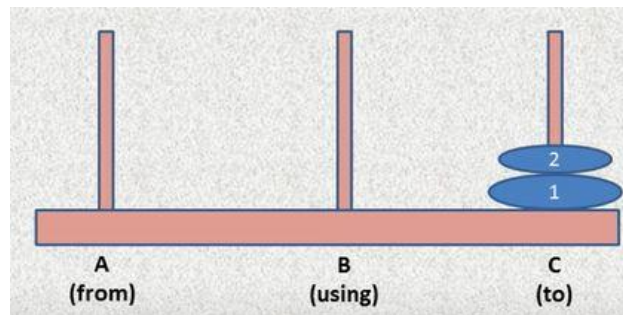

Fig a: Initial condition of disk

Step 1:

Fig1: Move a disk 2 from A to B

Step 2:

Fig2: Move a disk 1 from A to C

Step 3:

Fig3: Move a disk 2 from B to C

Example2:

2. Let us take an e.g. where number of disk (n) =3 and show the step by step process.

Solution:

Solution for 3 Disc
Fig a: Initial condition of disk
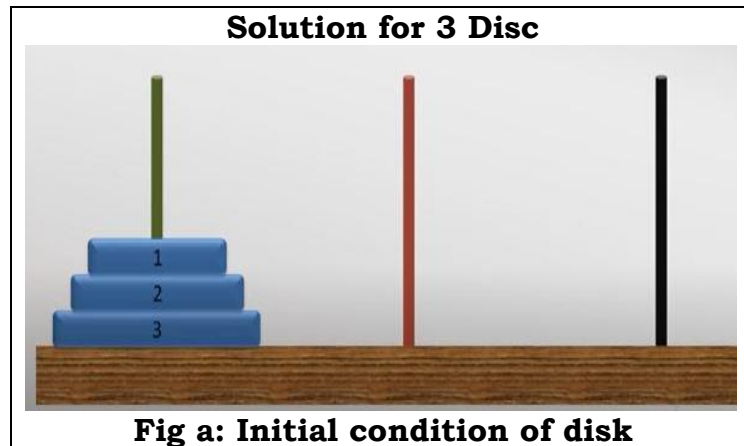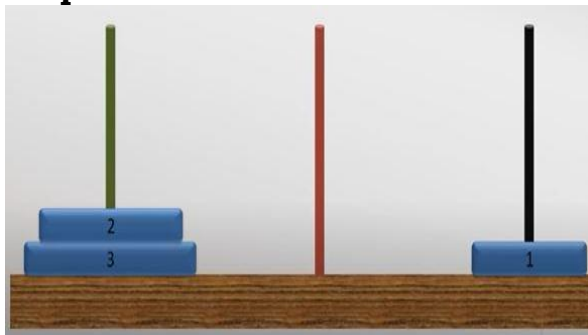
Step 1:	**Step 2:**
A B C	**A B C**
Fig1: Moving disk 1 from A to C	**Fig2: Moving disk 2 from A to B**
Step 3:	**Step 4:**
A B C	**A B C**
Fig3: Moving disk 1 from C to B	**Fig4: Moving disk 3 from A to C**
Step 5:	**Step 6:**
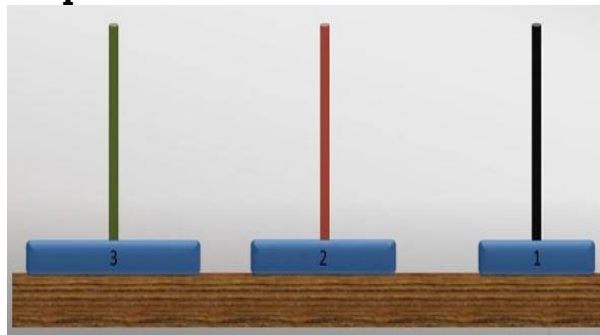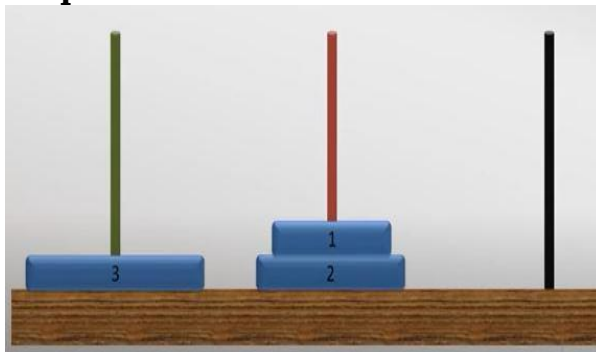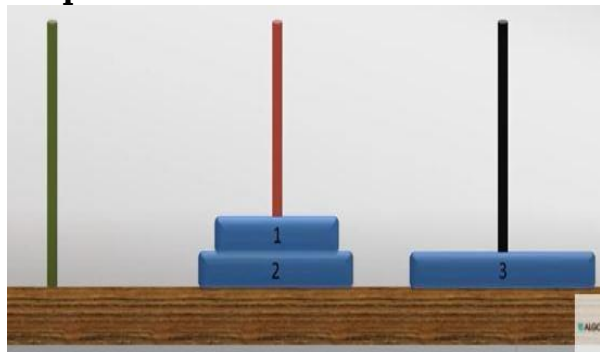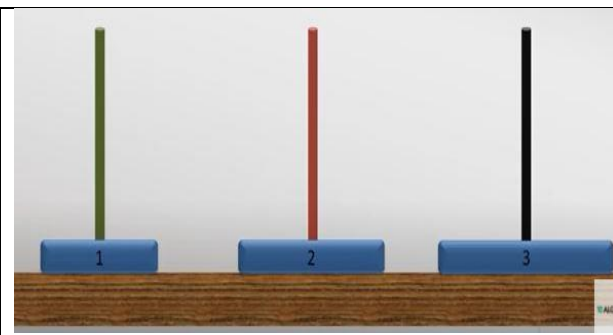
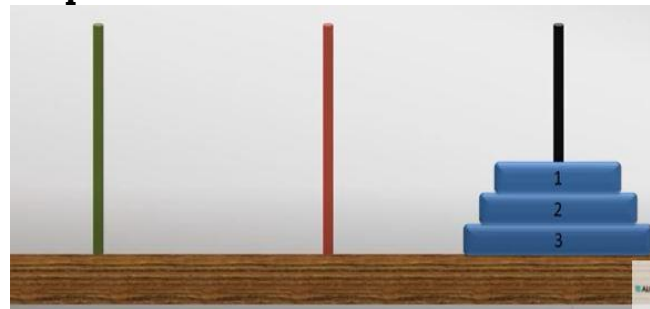**A**   **B**   **C**
Fig5: Moving disk 1 from B to A
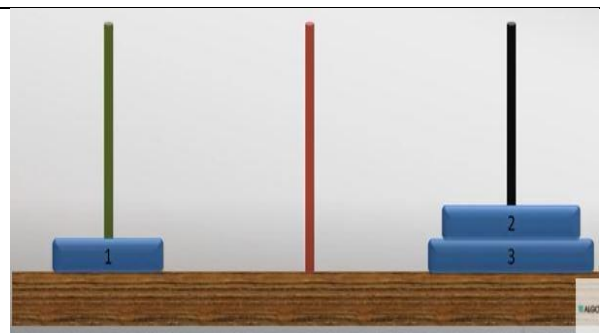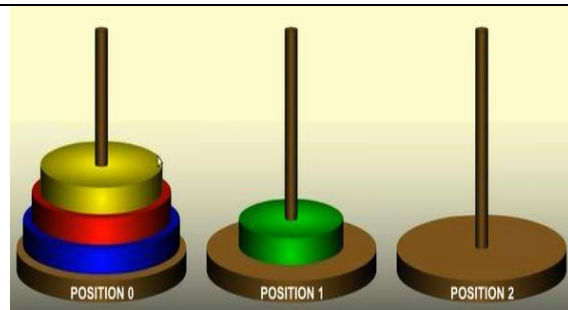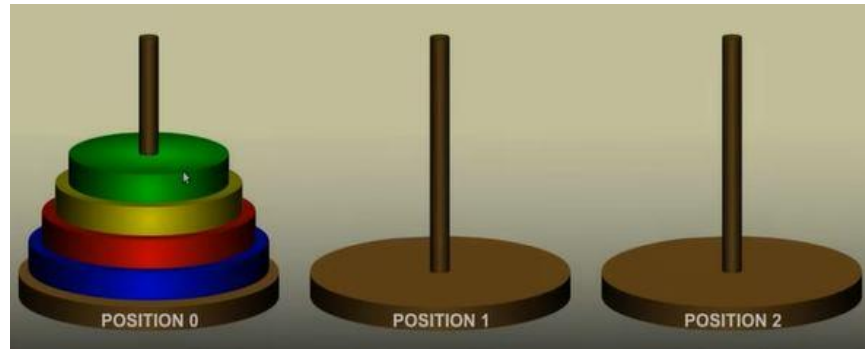

A   **B**   **C**
Fig6: Moving disk 2 from B to C
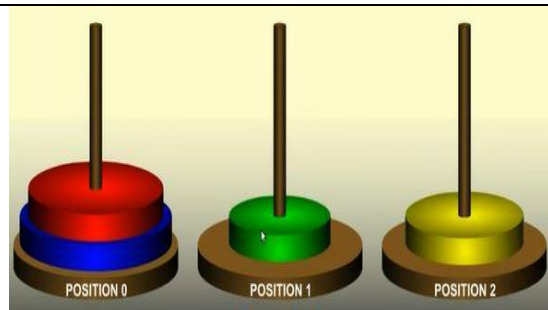
Step 7:

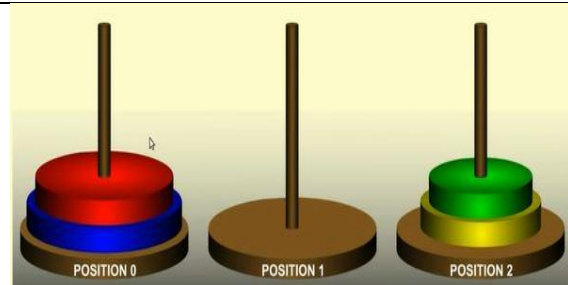**A**   **B**   **C**
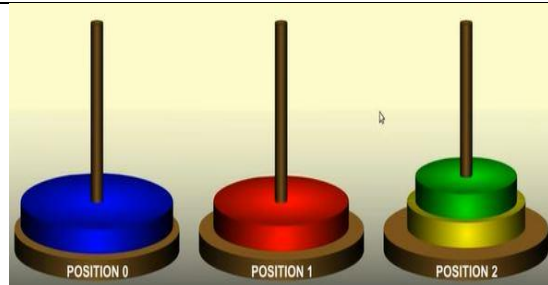**Fig7: Moving disk 1 from A to C**
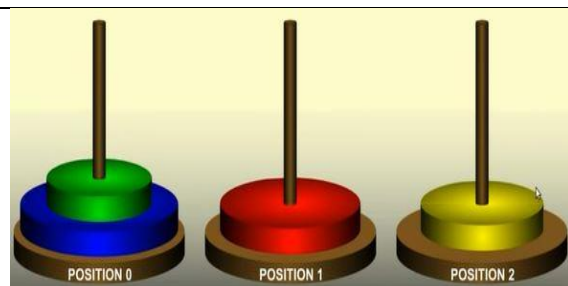
Example 3: for number of disk (n) =4
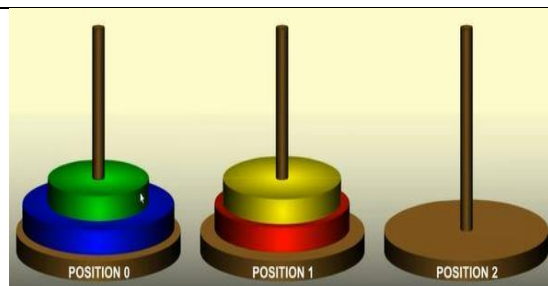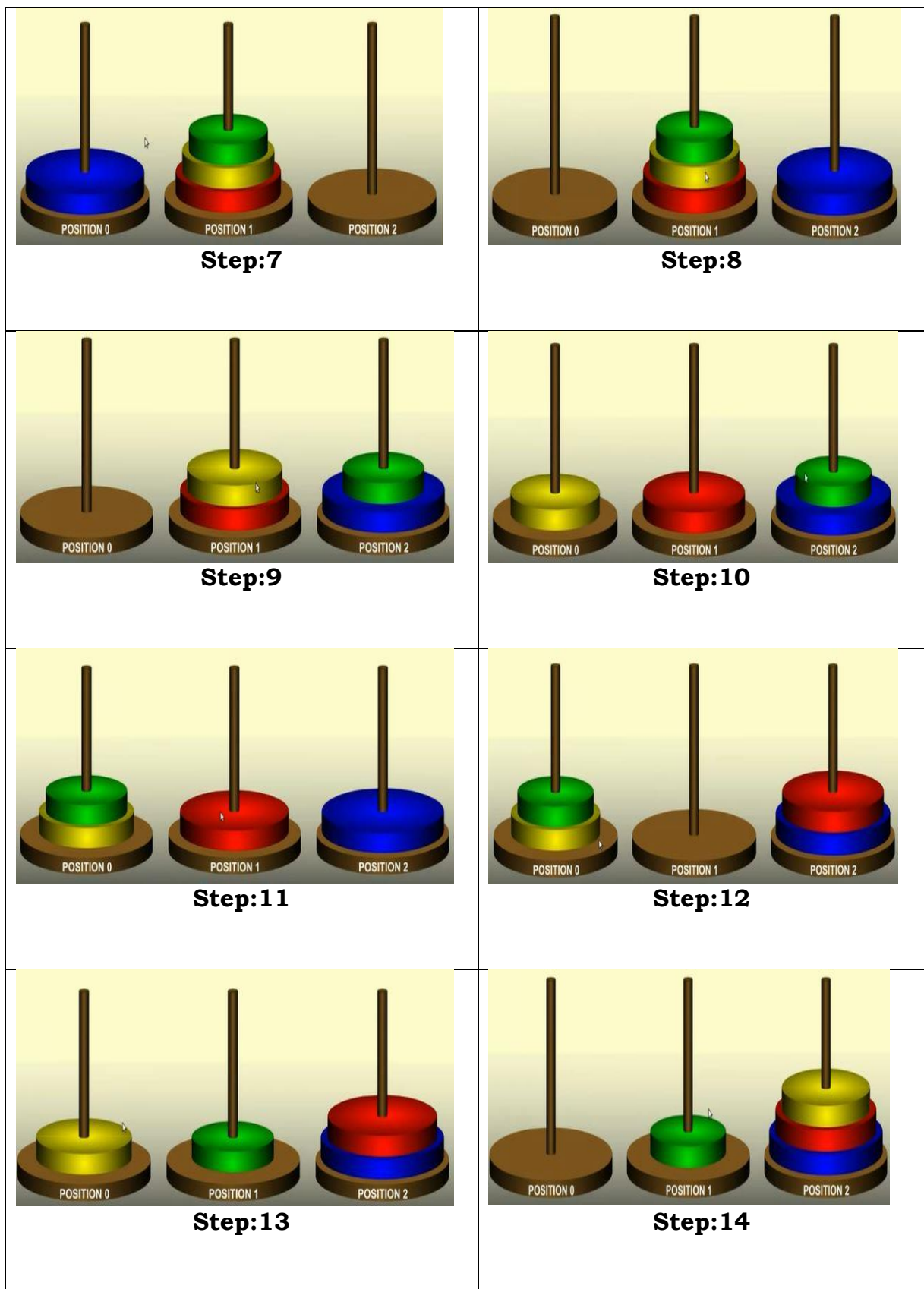
Step:1

Step:2

Step:3

Step:4

Step:5

Step:6

Step:7

Step:8

Step:9

Step:10

Step:11

Step:12

Step:13

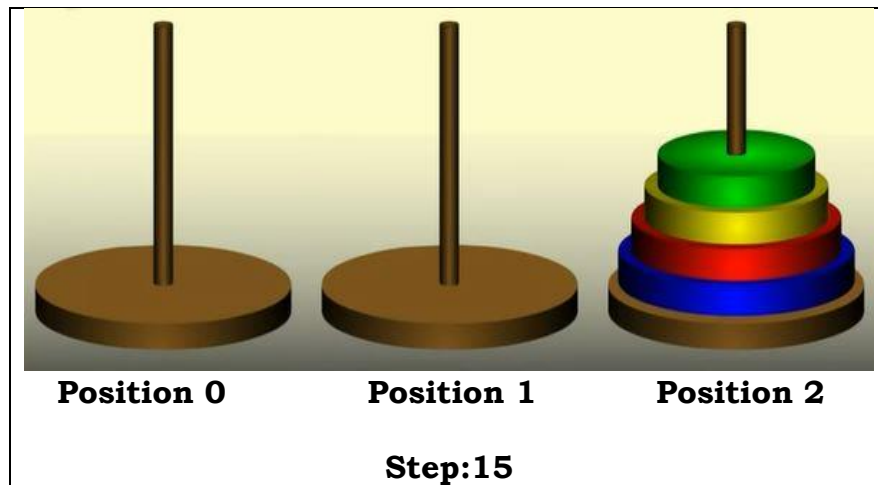Step:14

Position 0   **Position 1**   **Position 2**

Step:15

Program of TOH:

```c
// A C program for Tower of Hanoi using Recursion
#include<stdio.h>
#include<conio.h>

void towers(int, char, char, char);

int main()
{
    int num;
    clrscr();

    printf("Enter the number of disks : ");
    scanf("%d", &num);
    printf("The sequence of moves involved in the Tower of Hanoi are :\n");
    towers(num, 'A', 'C', 'B');
    getch();
    return 0;
}
void towers(int num, char frompeg, char topeg, char auxpeg)
{
    if (num == 1)
    {
     printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);
     return;
    }
    towers(num - 1, frompeg, auxpeg, topeg);
```

```
    printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);
    towers(num - 1, auxpeg, topeg, frompeg);
}
```

Output:

```
Enter the number of disks : 3
The sequence of moves involved in the Tower of Hanoi are :

 Move disk 1 from peg A to peg C
 Move disk 2 from peg A to peg B
 Move disk 1 from peg C to peg B
 Move disk 3 from peg A to peg C
 Move disk 1 from peg B to peg A
 Move disk 2 from peg B to peg C
 Move disk 1 from peg A to peg C_
```

Questions:
1. What is Recursion/Recursive Function?   2 marks
2. WAP to find the factorial of entered number using recursion. 5 marks
3. WAP to print Fibonacci series using recursive function.   5 marks
4. Explain the Tower of Hanoi problem with an example.   8 marks

Note:
➢ **Do above questions in a note copy.**
➢ **Make a lab report of TOH problem in a4 paper.**