# EXPERIMENT-1

## AIM:

Demonstrate recursive algorithms with examples.

(i) Factorial of a Number

(ii) Towers of Hanoi Problem

# DESCRIPTION:

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called a recursive function. A recursive function solves a particular problem by calling a copy of itself and solving smaller subproblems of the original problems. Many more recursive calls can be generated as and when required. It is essential to know that we should provide a certain case in order to terminate this recursion process. So we can say that every time the function calls itself with a simpler version of the original problem.

**1. Factorial of a given number:**

The factorial of a non-negative integer n, denoted by n!. It is the product of all positive integers less than or equal to n. For example, 5! =5\*4\*3\*2\*1=120

**Algorithm:**

Input: Integer ‘n’

Output: Factorial of ‘n’

Factorial(n)

{

  If n= 0 or 1 then

fact=1

else

fact= n\* factorial(n-1)

return(fact)

}

**Program:**

/\*This program returns factorial of a given number\*/

#include<stdio.h>

int factorial(int n){

if(n==0||n==1)

return 1;

else

return n\*factorial(n-1);

}

int main(){

int num;

printf("Enter the number:");

scanf("%d",&num);

int fact=factorial(num);

printf("Factorial:%d",fact);

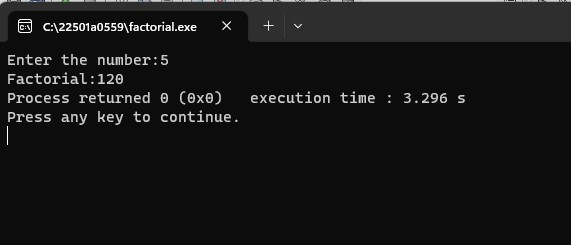
return 0;

}

**Output:**

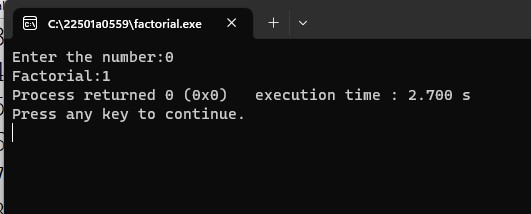
**Enter the number:5**

**Factorial:120**



**Enter the number:0**

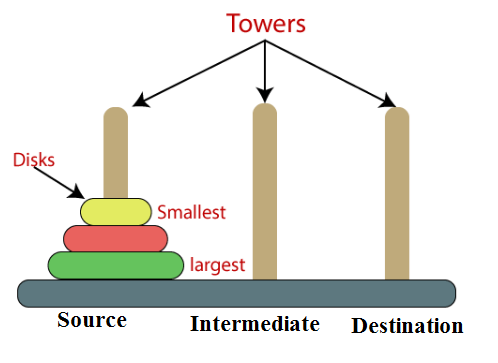
**Factorial:1**



**2. Towers of Hanoi Problem:**

The Tower of Hanoi is a mathematical game or puzzle. It consists of three rods, and a number of disks of different sizes which should be transferred from source to destination using the intermediate rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack to another rod, obeying the following rules:

* Only one disk may be moved at a time
* Each move consists of taking the upper disk from one of the rods and sliding into another rod, on the top of the other disks that may be already present on the rod.
* No disk may be placed on top of a smaller disk.

****

**Algorithm:**

Input: Integer ‘n’ number of disks

Output: ‘n’ disks are to be transferred from peg S (Source) to peg D (destination) with Peg I as the intermediate peg

1. Start

2. Read N value as the no. of disks

3. Call TOH(N, S, I, D).

4. Stop

TOH (N, S, I, D)

{

  if n = 1 then

Transfer disk from S to D and stop

Else

// transfer N-1 disks from peg S to peg I with peg D as the intermediate peg

Call TOH(N-1, S, D, I)

Transfer disk from S to D

// transfer N-1 disks from peg I to peg D with peg S as the intermediate peg

Call TOH(N-1, I, S, D);

}

**Program:**

/\*Towers of hanoi\*/

#include <stdio.h>

int toh(int n,int src,int dest,int aux){

if(n>=1){

toh(n-1,src,aux,dest);

printf("move disk %d from rod %d to rod %d\n",n,src,dest);

toh(n-1,aux,dest,src);

}

}

int main() {

int N;

printf("Enter the number of disks:");

scanf("%d", &N);

toh(N, 1, 3, 2);

return 0;

}

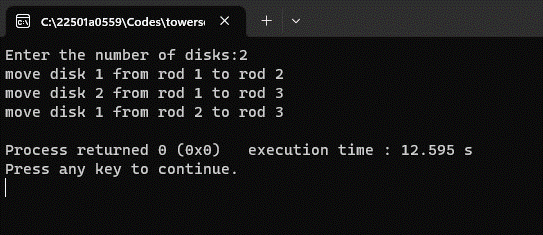
**Output:**

**Enter the number of disks:2**

**move disk 1 from rod 1 to rod 2**

**move disk 2 from rod 1 to rod 3**

**move disk 1 from rod 2 to rod 3**



**Enter the number of disks:4**

**move disk 1 from rod 1 to rod 2**

**move disk 2 from rod 1 to rod 3**

**move disk 1 from rod 2 to rod 3**

**move disk 3 from rod 1 to rod 2**

**move disk 1 from rod 3 to rod 1**

**move disk 2 from rod 3 to rod 2**

**move disk 1 from rod 1 to rod 2**

**move disk 4 from rod 1 to rod 3**

**move disk 1 from rod 2 to rod 3**

**move disk 2 from rod 2 to rod 1**

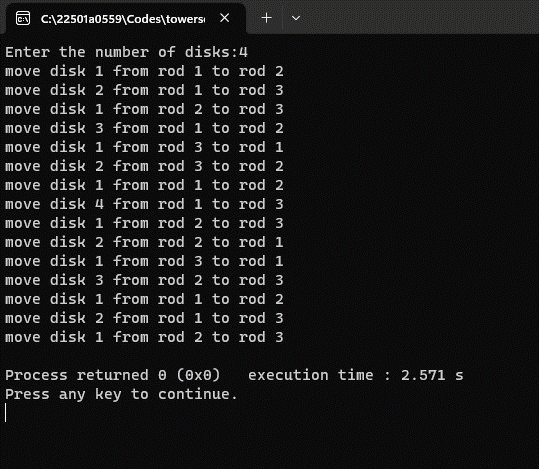
**move disk 1 from rod 3 to rod 1**

**move disk 3 from rod 2 to rod 3**

**move disk 1 from rod 1 to rod 2**

**move disk 2 from rod 1 to rod 3**

**move disk 1 from rod 2 to rod 3**



**3. Program to perform Binary Search using Recursion.**

**Program:**

/\*Binary search with recursion\*/

#include <stdio.h>

int binarySearch(int arr[], int low, int high, int x)

{

if (high >= low){

int mid = (low+high)/2;

if (arr[mid] == x) return mid;

if (arr[mid] > x) return binarySearch(arr, low, mid-1, x);

return binarySearch(arr, mid+1, high, x);

}

return -1;

}

int main(void)

{

int arr[200];

int n,x;

printf(“Enter the number of elements:\n”);

scanf(“%d”,&n);

printf(“Enter the elements:\n”);

for(int i=0;i<n;i++){

scanf(“%d”,&arr[i]);

}

printf(“Enter the value to be searched:\n”);

scanf(“%d”,&x);

int result = binarySearch(arr, 0, n-1, x);

if (result == -1) {

printf("Element is not present in array");

}else{

printf("Element is present at index %d", result);

}

return 0;

}

**Output:**

**Enter the number of elements:**

**5**

**Enter the elements:**

**4**

**3**

**2**

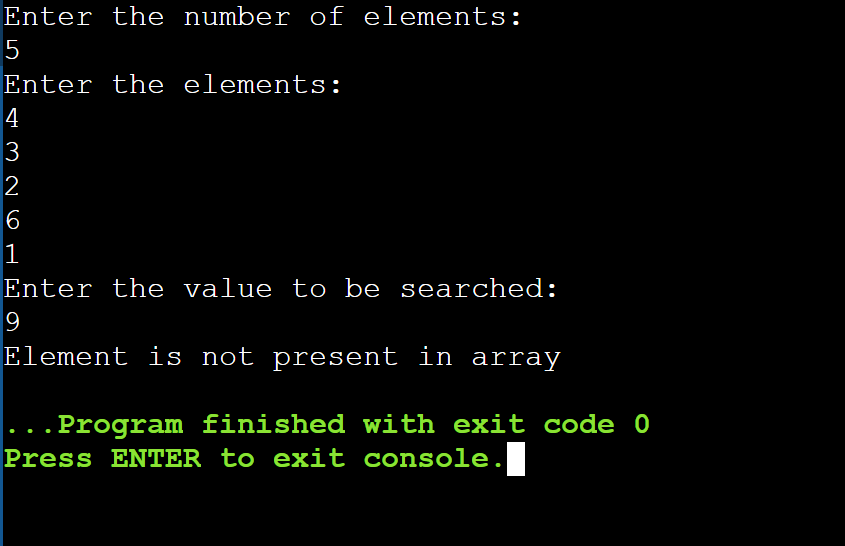
**6**

**1**

**Enter the value to be searched:**

**9**

**Element is not present in array**

****

**Enter the number of elements:**

**4**

**Enter the elements:**

**1**

**2**

**3**

**4**

**Enter the value to be searched:**

**3**

**Element is present at index 2**

