# DATA STRUCTURES

# E DEEPAK

# 192425197

# EXP-1 Matrix Multiplication

## Aim

To write a C program to perform matrix multiplication.

## Algorithm

1. Start.
2. Read the dimensions and elements of matrix A and B.
3. Check if multiplication is possible (columns of A = rows of B).
4. Perform multiplication using triple nested loop.
5. Store result in matrix C.
6. Display the resultant matrix.
7. Stop.

A screenshot of a computer

AI-generated content may be incorrect.

## Result

The program was successfully executed and the product of two matrices was obtained.

# EXP-2 Odd or Even Numbers

## Aim

To write a C program to find odd or even numbers from a given set of numbers.

## Algorithm

1. Start.
2. Read the number of elements.
3. Read the set of numbers.
4. Check each number using modulus operator %2.
5. Print whether it is Odd or Even.
6. Stop.

A screenshot of a computer

AI-generated content may be incorrect.

## C Program

## Result

The program was successfully executed and the given set was classified as odd or even.

# EXP-3 Factorial without Recursion

## Aim

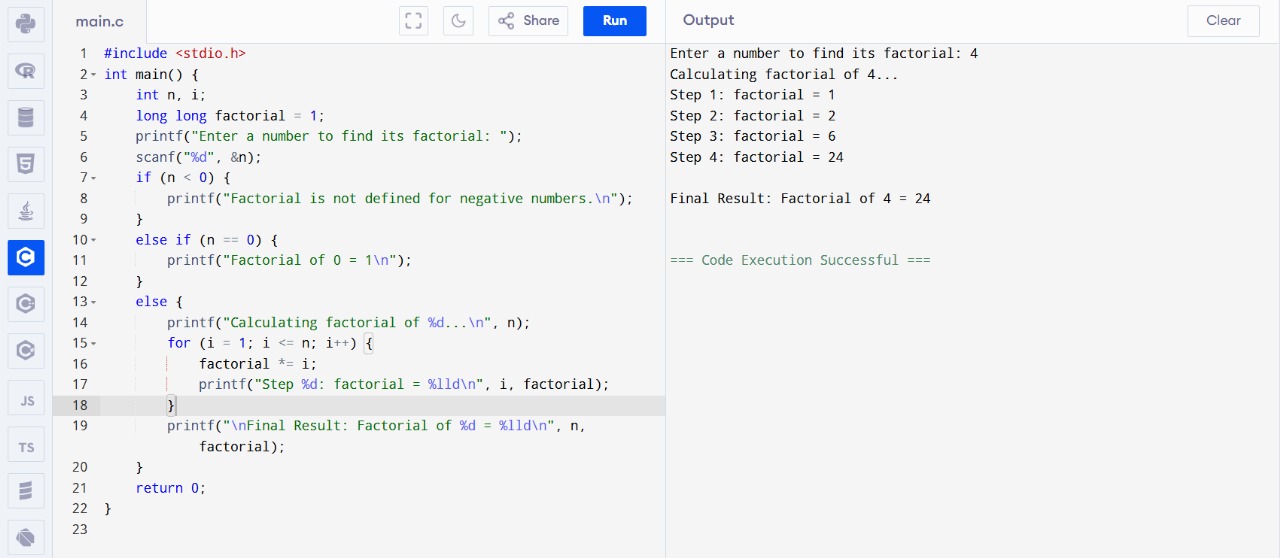
To write a C program to find the factorial of a given number without using recursion.

## Observation

Factorial n! = n × (n-1) × ... × 1.

## Algorithm

1. Start.
2. Read integer number n.
3. Initialize fact=1.
4. Multiply fact by numbers from 1 to n.
5. Print fact.
6. Stop.



## C Program

## Result

The program was successfully executed and factorial was found without recursion.

# EXP-4 Fibonacci without Recursion

## Aim

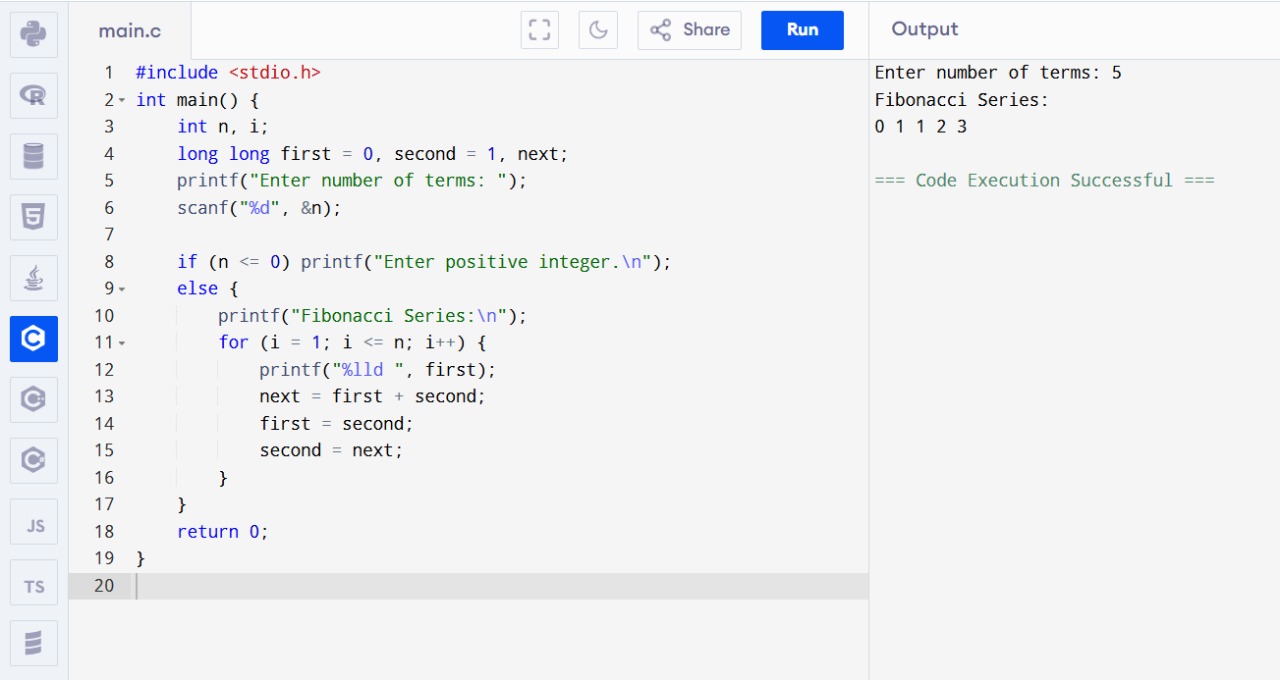
To write a C program to print Fibonacci series without recursion.

## Observation

Fibonacci sequence: 0,1,1,2,3,... where f(n)=f(n-1)+f(n-2).

## Algorithm

1. Start.
2. Read number of terms n.
3. Initialize first two terms 0 and 1.
4. Use loop to compute next terms.
5. Display series.
6. Stop.



## Result

The program was successfully executed and Fibonacci series was generated without recursion.

# EXP-5 Factorial using Recursion

## Aim

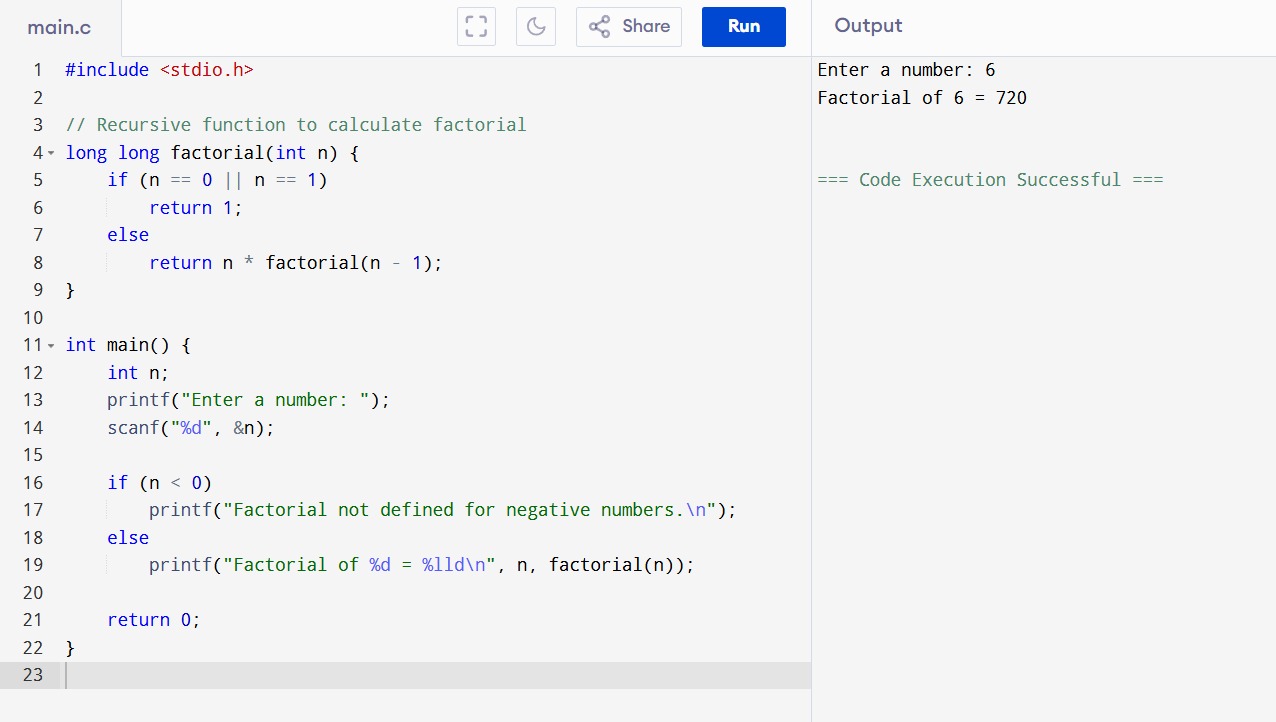
To write a C program to find the factorial of a given number using recursion.

## Observation

Factorial of a number n is defined as n! = n × (n-1) × ... × 1.

## Algorithm

1. Start.
2. Read integer number n.
3. Define recursive function fact(n).
4. If n==0 or n==1 return 1 else return n\*fact(n-1).
5. Call function and print result.
6. Stop.



## Result

The program was successfully executed and factorial was found using recursion.

# EXP-6 Fibonacci using Recursion

## Aim

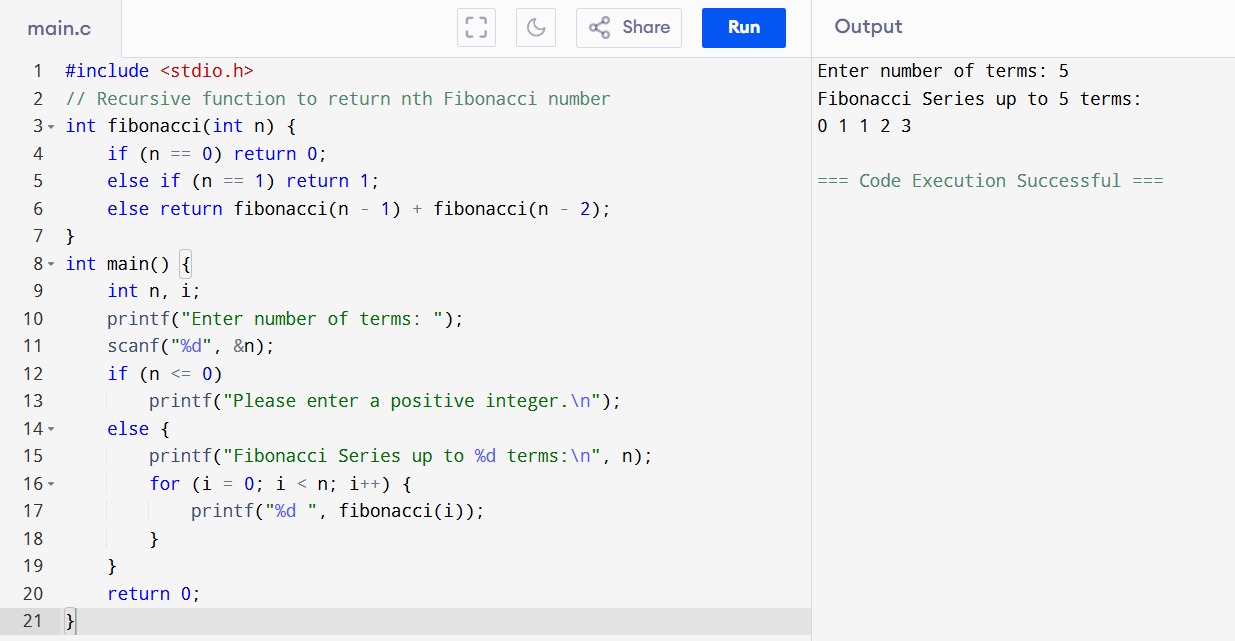
To write a C program to print Fibonacci series using recursion.

## Observation

Fibonacci sequence is generated using recursive relation f(n)=f(n-1)+f(n-2).

## Algorithm

1. Start.
2. Read n.
3. Define recursive function fib(n).
4. If n==0 return 0, if n==1 return 1.
5. Else return fib(n-1)+fib(n-2).
6. Call function for first n terms.
7. Display result.



## Result

The program was successfully executed and Fibonacci series was generated using recursion.

# EXP-7 Array Operations (Insert, Delete, Display)

## Aim

To write a C program to implement array operations such as insert, delete, and display.

## Observation

Arrays are fixed-size collections. We can insert, delete and display elements using index-based operations.

## Algorithm

1.Start.

2.Declare array and size.

3.Insert element at given position.

4.Delete element from given position.

5.Display array elements.

6.Stop.

## C Program

## 

## Result

The program was successfully executed and array operations were performed.

# Linear Search

## Aim

To write a C program to search a number using Linear Search method.

## Observation

Linear search compares each element until key is found.

## Algorithm

1. Start.
2. Read array elements and key.
3. Scan array sequentially.
4. If element matches key, print position.
5. If not found, print not found.
6. Stop.

## C Program

## Result

The program was successfully executed and the element was searched using Linear Search.

# Binary Search

## Aim

To write a C program to search a number using Binary Search method.

## Algorithm

1. Start.
2. Read sorted array and key.
3. Initialize low=0, high=n-1.
4. While low<=high compute mid.
5. If key==a[mid], found.
6. Else if key<a[mid], high=mid-1 else low=mid+1.
7. Stop.

## C Program

## Sample Input and Output

Input:  
5  
10 20 30 40 50  
Key=40  
  
Output:  
Element found at position 4

## Result

The program was successfully executed and the element was searched using Binary Search.