public class sumDigits {

public static int sumOfDigits(int num){

// base case condition

if(num == 0){

return 0;

}

// recursive function call

return (num % 10) + sumOfDigits(num/10);

}

public static void main(String[] args){

int num = 1234;

int result = sumOfDigits(num);

System.out.println(result);

}

}

class factorialFinding {

public static int factorialFinding(int n){

int result = 0;

// base case condition

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

return 1;

}

else{

// recursive function call

result = n \* factorialFinding(n-1);

}

return result;

}

public static void main(String[] args) {

int num = 5;

int result = factorialFinding(num);

System.out.println("Factorial of a given number is: "+result);

}

}

1. Climbing Stairs

Optimized Approach

// time complexity: O(n)

// space complexity: O(1)

// Iterative approach

class Solution {

public int climbStairs(int n) {

if(n == 1){

return 1;

}

int num1 = 1;

int num2 = 2;

for(int i=3; i<=n; i++){

int num3 = num1 + num2;

num1 = num2;

num2 = num3;

}

return num2;

}

}

2. Fibonacci Number

// time complexity: O(2^n)

// space complexity: O(n)

class Solution {

public int fib(int n) {

// base case condition

if(n <= 1){

return n;

}

// recursive function call

return fib(n-1) + fib(n-2);

}

}