Lab#10

Recursion

Objective

- To understand the basic concepts of Recursion.

Theory

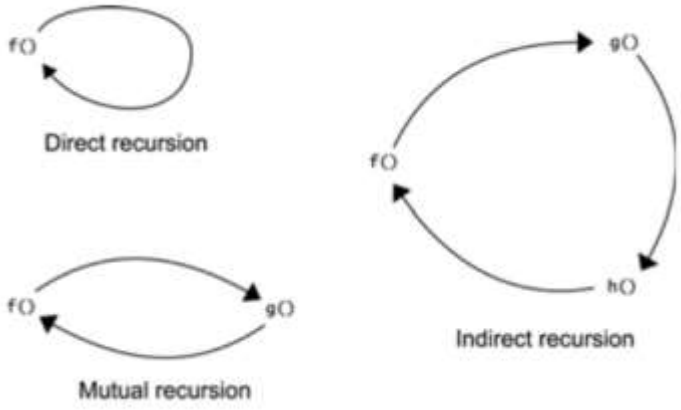
Recursion

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily.

Recursion has many, many applications. In this module, we'll see how to use recursion to compute the factorial function, to determine whether a word is a palindrome, to compute powers of a number, to draw a type of fractal, and to solve the ancient Towers of Hanoi problem. Later modules will use recursion to solve other problems, including sorting.

Types of Recursions:

1. Direct Recursion
2. Mutual Recursion
3. Indirect Recursion



Applications of Recursion:

1. Chess Games

Happy Coding 😊

2. Factorial Number
3. Fibonacci series

Recursion example: Factorial of an n number

For $n = 1$, $1! = n! = n(n-1)! = 1(1-1)! = 1(0)! = 1(1) = 1$.
For $n = 2$, $2! = n! = n(n-1)! = 2(2-1)! = 2(1)! = 2(1) = 2$.
For $n = 3$, $3! = n! = n(n-1)! = 3(3-1)! = 3(2)! = 3(2) = 6$.
For $n = 4$, $4! = n! = n(n-1)! = 4(4-1)! = 4(3)! = 4(6) = 24$.
For $n = 5$, $5! = n! = n(n-1)! = 5(5-1)! = 5(4)! = 5(24) = 120$.

Here is a trace of the call `f(5)` to the recursive factorial function defined in Example 9.2:

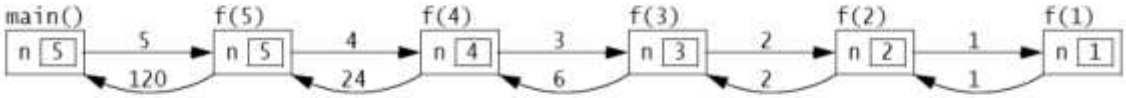


Figure 9.1 Tracing the recursive factorial function

Code Example

```
public class testing {
    static int fac(int n){
        if(n<=0)
            return 1;
        else{
            return n*fac(n-1);
        }
    }
    static void RecFun(int n){
        if(n<0)
            return;
        else{
            System.out.println("Number : "+n);
            RecFun(n-1);
        }
```

Happy Coding 😊

```
    }
    public static void main(String[] args) {
    //  RecFun(5);
        System.out.println("Fact = "+fac(4));
    }
}
//fac of number 3 -> 4x3x2x1=6
```

Output:

```
Number : 5
Number : 4
Number : 3
Number : 2
Number : 1
Number : 0
Fact = 24
PS C:\Users\Hp\Downloads\Rafia\Rafia\DSA\Labs\
```

Exercise

Tasks

1. Write a program to find the factorial of a number using recursion. Take input from user.

$$n! = \begin{cases} 1, \text{if } n = 0 \\ n(n-1)!, \text{if } n > 0 \end{cases}$$

2. Write a program to use recursion as a loop to print numbers 1 to n. Take n as input from user.

3. Write a java program that generate Fibonacci series.

The *Fibonacci numbers* are 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... Each number after the second is the sum of the two preceding numbers. This is a naturally recursive definition:

$$F_n = \begin{cases} 0, \text{if } n = 0 \\ 1, \text{if } n = 1 \\ F_{n-1} + F_{n-2}, \text{if } n > 1 \end{cases}$$

Happy Coding 😊

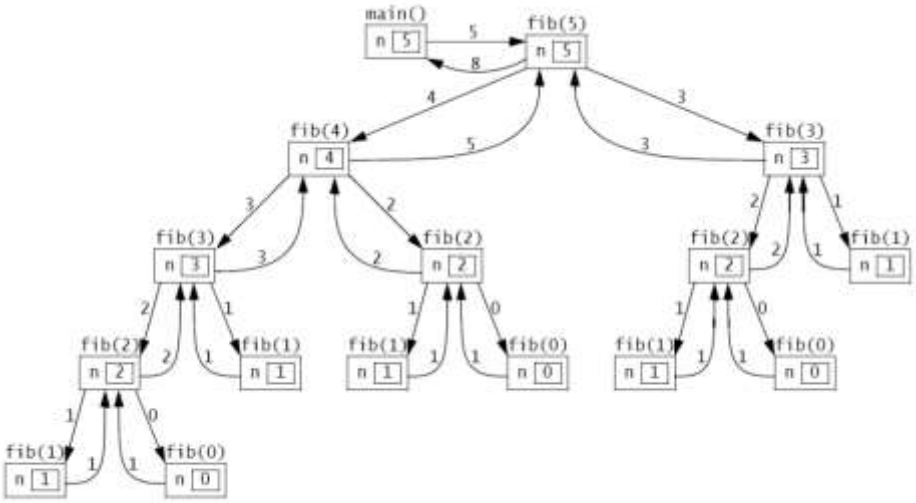


Figure 9.2 Tracing the recursive Fibonacci function

4. Write a java program that implements binary search using recursive technique.

Here is the *recursive binary search algorithm*:
(Precondition: $s = \{s_0, s_1, \ldots, s_{n-1}\}$ is a sorted sequence of n ordinal values of the same type as x.)
(Postcondition: either the index i is returned where $s_i = x$, or -1 is returned.)

1. If the sequence is empty, return -1.
2. Let s_i be the middle element of the sequence.
3. If $s_i = x$, return its index i.
4. If $s_i < x$, apply the algorithm on the subsequence that lies above s_i.
5. Apply the algorithm on the subsequence of s that lies below s_i.

Happy Coding 😊