Chapter 5 :

Computer Science

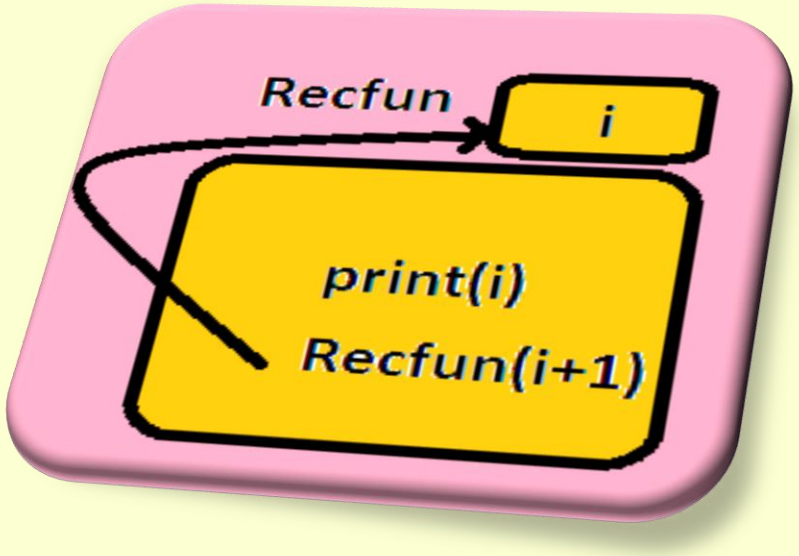
Class XII (As per CBSE Board)

New Syllabus 2019-20

Visit : python.mykvs.in for regular updates

Recursion

It is a way of programming or coding technique, in which a function calls itself for one or more times in its body. Usually, it is returning the return value of this function call procedure. If a function definition fulfils such conditions, we can call this function a recursive function.



Recursion

The Two Laws of Recursion

- **Must have a base case** - There must be at least one base criteria/condition, when such condition is met the function stops calling itself.
- **Must move toward the base case** - The recursive calls should moves in such a way that each time it comes closer to the base criteria.

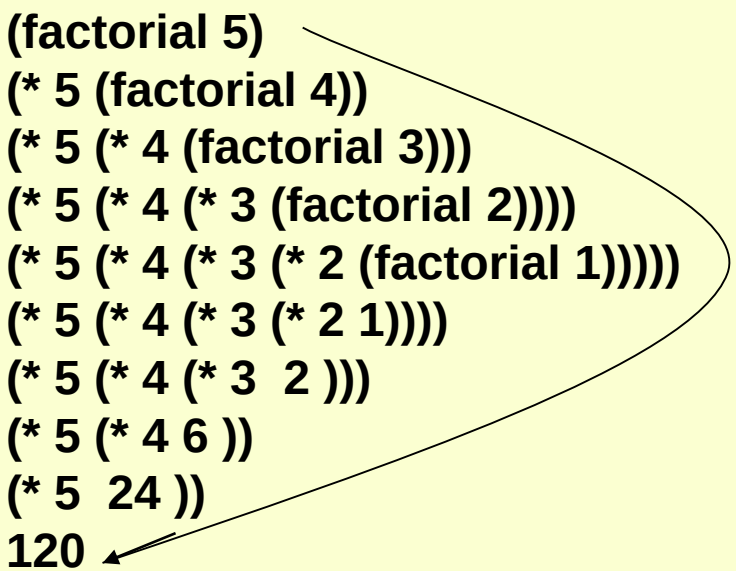
Recursion

Factorial of a Number Using Recursion

ALGORITHM

1.Test if n <= 0. If so, return 1.

2.If not, then call the factorial algorithm with n – 1 and multiply the result by n and return that value.



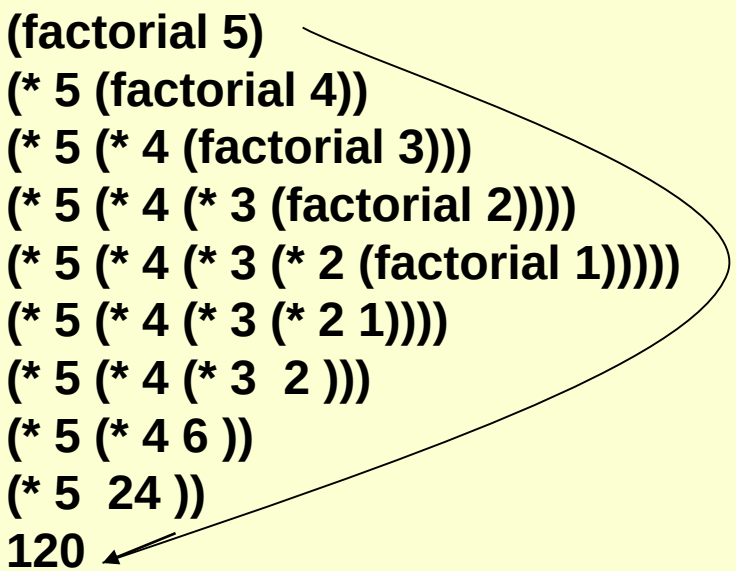
Recursion

Factorial of a Number Using Recursion

PYTHON PROGRAM

```
def factorial(x):
    if x==1:
        return 1
    else:
        return x*factorial(x-1)

f=factorial(5)
print ("factorial of 5 is ",f)
```



Recursion

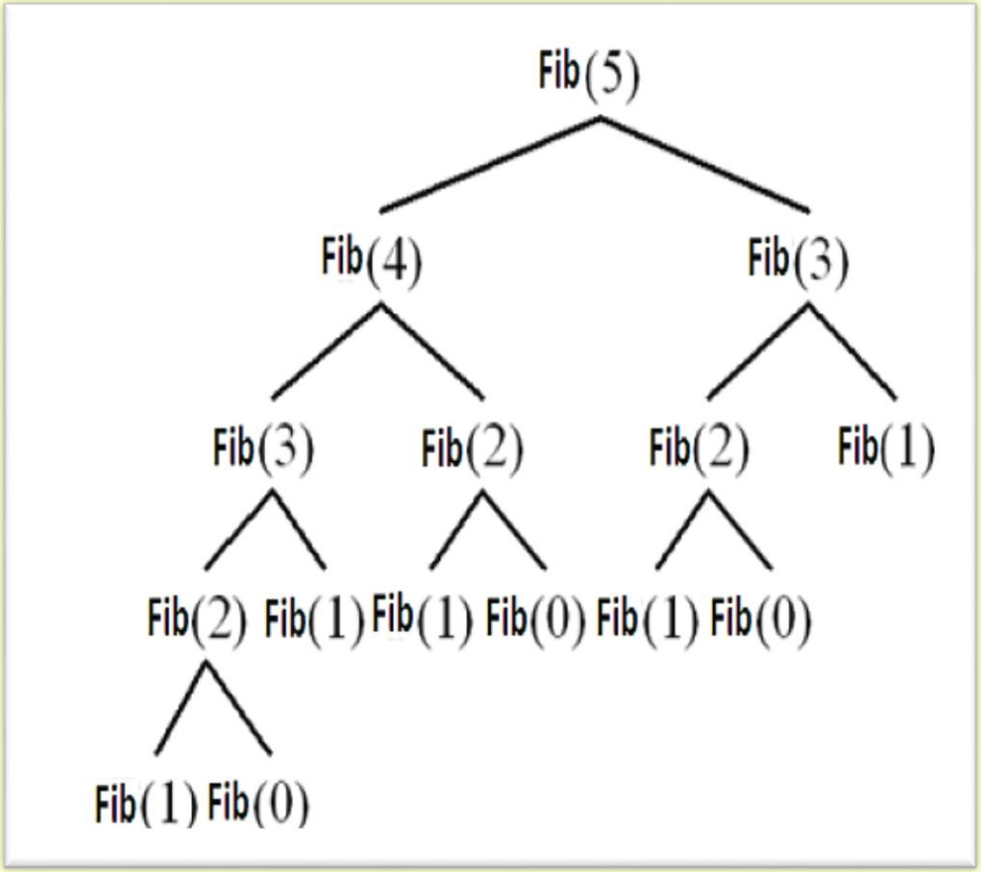
Fibonacci numbers Using Recursion

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, …

Algorithm

Fib(n)

1. If n =1 or n=2, then
2. return 1
3. Else
4. a = Fib(n-1)
5. b = Fib(n-2)
6. return a+b



Recursion

Fibonacci numbers Using Recursion

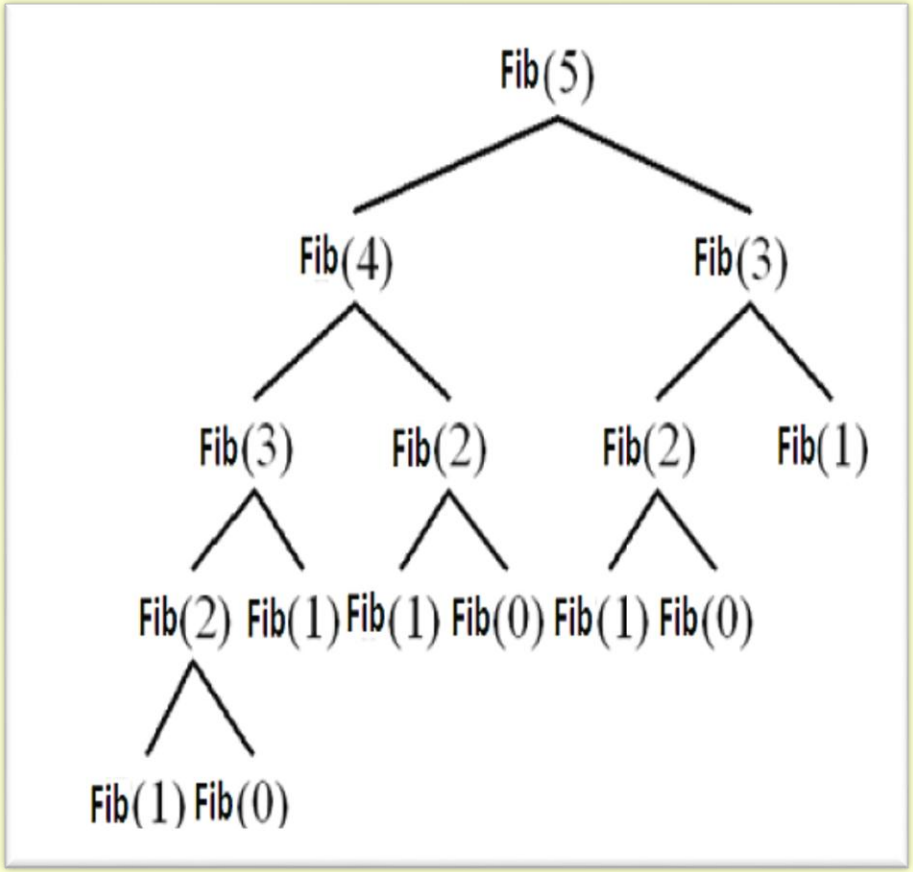
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, …

Program

```
def fib(n):
   if n <= 1:
      return n
   else:
      return(fib(n-1) + fib(n-2))

nterms = int(input("enter a number"))

if nterms <= 0:
   print("Plese enter a positive integer")
else:
   print("Fibonacci sequence:")
   for i in range(nterms):
      print(fib(i))
```



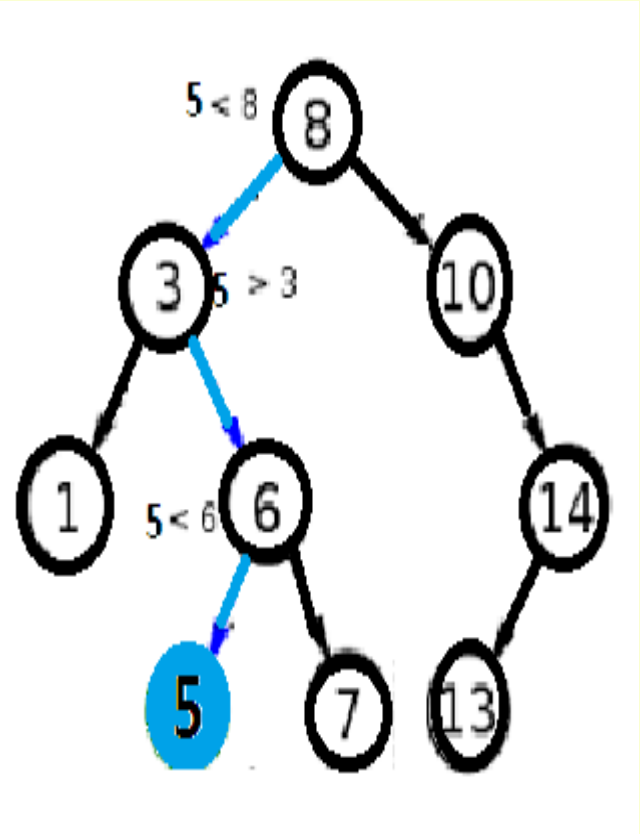
Recursion

Binary Search Using Recursion

Algorithm

1. Find the midpoint of the array; this will be the element at arr[size/2]. The midpoint divides the array into two smaller arrays: lower half and upper half
2. Compare key to arr[midpoint] by calling the user function cmp_proc.
3. If the key is a match, return arr[midpoint]; otherwise
4. If the array consists of only one element return NULL, indicating that there is no match; otherwise
5. If the key is less than the value extracted from arr[midpoint] search the lower half of the array by recursively calling search; otherwise
6. Search the upper half of the array by recursively calling search.

NOTE:- For binary search all elements must be in order.



Recursion

Binary Search Using Recursion

Program

```
def binarySearch (arr, first, last, x):
    if last >= first:
        mid =int( first + (last - first)/2)
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binarySearch(arr, first, mid-1, x)
        else:
            return binarySearch(arr, mid+1, last, x)
    else:
        return -1

arr = [ 1,3,5,6,7,8,10,13,14 ]
x = 10
result = binarySearch(arr, 0, len(arr)-1, x)

if result != -1:
    print ("Element is present at index %d" % result)
else:
    print ("Element is not present in array")
```