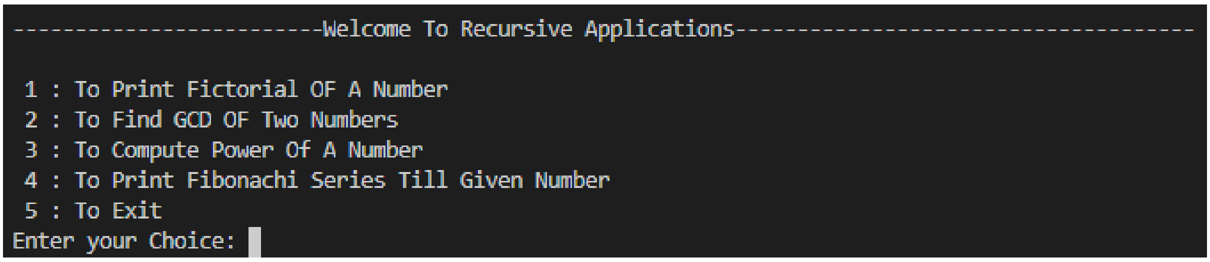
Recursion

Definition:

A recursive function is a function that calls itself during its execution. It depend on its previous values to create new ones

Project Details:

Our project is totally based on recursion. We implemented the following algorithms in python. User will be asked to enter which algorithm he wants to run. The one he will choose, will be executed.



```
-------------------------Welcome To Recursive Applications-------------------------------------

 1 : To Print Fictorial OF A Number
 2 : To Find GCD OF Two Numbers
 3 : To Compute Power Of A Number
 4 : To Print Fibonachi Series Till Given Number
 5 : To Exit
Enter your Choice: 
```

1. FACTORIAL NUMBER:

ALGORITHM:

```
Factorial(n)        [where n is a positive integer]
if  n = 0
        return 1
else
        return n.Factorial (n − 1)
```

FLOW:

Factorial(5)
5. Factorial(4)
5.4. Factorial(3)
5.4.3. Factorial(2)
5.4.3.2. Factorial(1)
5.4.3.2.1

EXPLANATION:

Let Suppose 6 passed as an argument.

Then, 5 is passed to Factorial () from the same function (recursive call). In each recursive call, the value of argument n is decreased by 1.

When the value of n is less than 1, there is no recursive call and the factorial is returned ultimately to the function.

2. GCD:

ALGORITHM:

```
Gcd(a, b)      [where a, b must be positive & a < b]
if  a = 0
        return b
else
        return  gcd (b mod  a, a)
```

FLOW:

Gcd (50, 3)
Gcd(50mod3,3)
Gcd(2,3)

Gcd(3,2)
Gcd(3mod2,2)
Gcd(1,2)

EXPLANATION:

In the above algorithm, Gcd() is a recursive function. It has two parameters i.e. a and b. If b is greater than 0, then a is returned to the main() function. Otherwise, the Gcd() function recursively calls itself with the values b and a%b.

3. POWER:

ALGORITHM:

```
Power(a, n)  [where a must be real number, a>0 & n is a positive integer]
if  n = 0
        return 1
else
        return a.Power (a, n − 1)
```

FLOW:

Power(5, 3)
5.Power(5, 2)
5.5.Power(5,1)
5.5.5

EXPLANATION:

In the above algorithm, the function Power() is a recursive function. If the power is zero, then the function returns 1 because any number raised to power 0 is 1. If the power is not 0, then the function recursively calls itself.

4. FIBONACCI SERIES:

ALGORITHM:

```
int F(int n)
if n <=2
          return n
 else
          return F(n-1) + F(n-2)
```

FLOW (Only Left hand side of a tree like structure):

F(6)
F(6-1)+ F(6-2)
F (5) + F (4)
=5+4=8

F(5)
F(5-1) + F(5-2)
F4) + F(3)
=3+2=5

F(4)
F(4-1)+ F (4-2)
F(3)+ F(2)
=2+1=3

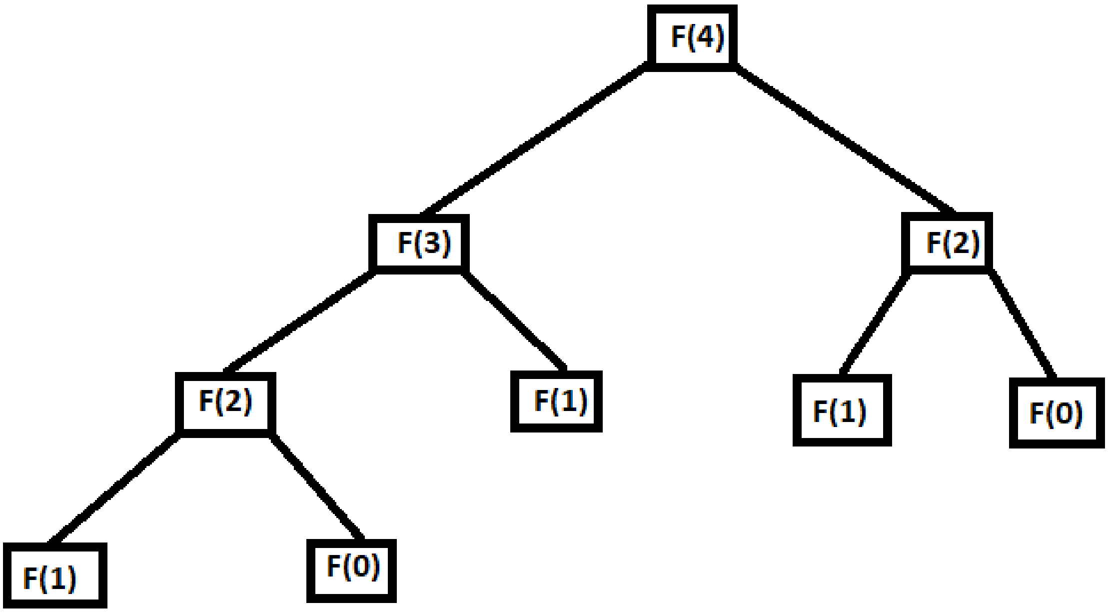
F(4)
F(4-1) + F(4-2)
F(3) + F(2)
=2+1=3

F(3)
F(3-1) + F(3-2)
F(2) + F(1)
=1+1=2

F(3)
F(3-1) + F(3-2)
F (2)+ F (1)
=1+1=2

F(2)
F(2-1) + F(2-2)
F(1)
=1

Tree with explanation:



Let suppose an example of F(4). Which requires two recursive call to F(3) & F(2). Notice how inefficient this is, just to compute F(4) we needed nine function calls. Or you can say this is a classic example of when not to use recursion.