16.1 Recursion

Recursion

when the function calls itself to make problem smaller.

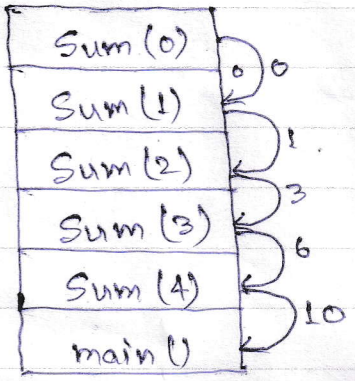
Q Add all numbers till n

Sum till n = n + n-1 + n-2 + n-3 ..... + 1
Sum till n = n + sum till n-1
Sum till n-1 = n-1 + sum till n-2
⋮ ⋮
Sum till 0 = 0

Code:

```
int Sum (int n) {
    int prevSum = Sum(n-1);
    if (n==0) {                    // Break point // base condition
        return 0;
    }
    int prevSum = Sum(n-1);
    return n + prevSum;
}
```

n = 4



Sum = 10

Q Calculate n raised to power of p

$n^p = n * n * n * n * n$ ..... p times

$n^p = n * n^{p-1}$

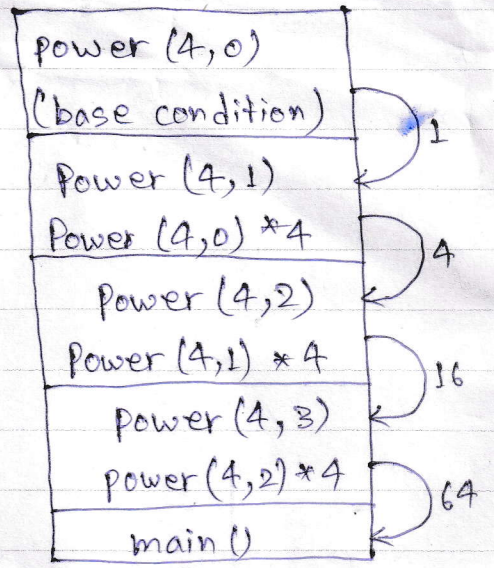
$n^0 = 1$ // base condition

Code:

```
int power (int n , int p) {

    if (p == 0) {
        return 1;
    }
    int prevPower = power (n , p-1);
    return n * prevPower ;
}
```

power (4, 3).



power (4,3) = 64

Q Find the factorial of a number n

$n! = n * n-1 * n-2 * \ldots * 1$

$n! = n * (n-1)!$

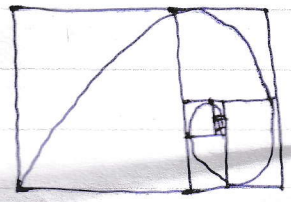
$0! = 1$ // base case

Code:

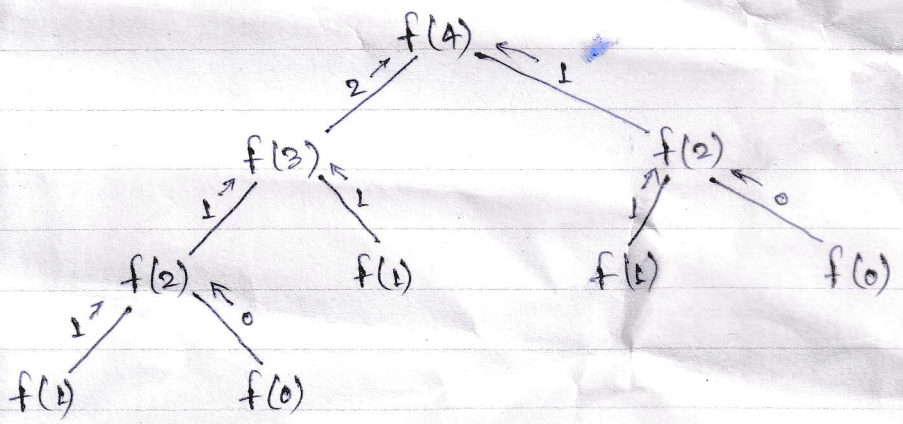
```
int factorial (int n) {

    if (n == 0) {
        return 1;
    }
    int prevfact = factorial (n-1);
    return n * prevfact;
}
```

Q Print the nth Fibonacci number



0, 1, 1, 2, 3, 5, 8, 13, ...
1 2 3 4 5 6 7



$fib(n) = fib(n-1) + fib(n-2)$

$fib(0) = 0$ $fib(1) = 1$

Code:

```
int fib (int n) {

    if (n==0 || n==1) {          // base case
        return n;
    }
    return fib(n-1) + fib(n-2);
}
```