1) Define Recursion

2) How write recursion

3) How it works

4) TC / SC of recursion

Recursion

⇒ A function calling itself

⇒ When a problem can be solved by solving smaller occurances of the same problem

⇒ Subproblems

Eg Calculate sum of first N natural numbers.

$$Sum(N) = 1 + 2 + 3 + 4 \ldots\ldots N-2 + N-1 + N$$

$$Sum(N) = Sum(N-1) + N$$

Subproblem

Steps

1) Assumption

⇒ Decide what your function does
⇒ Assume it returns correct answer

2) Main Logic

⇒ Solve the original problem using the sol^{n} of the subproblem

3) Base Case

⇒ When should your recursion stop.

Sum (N) using recursion

⇒ int sum (N) {

// Assumption : Returns the correct sum of the first N natural no.

// Base Case

if (N == 1) return (1);

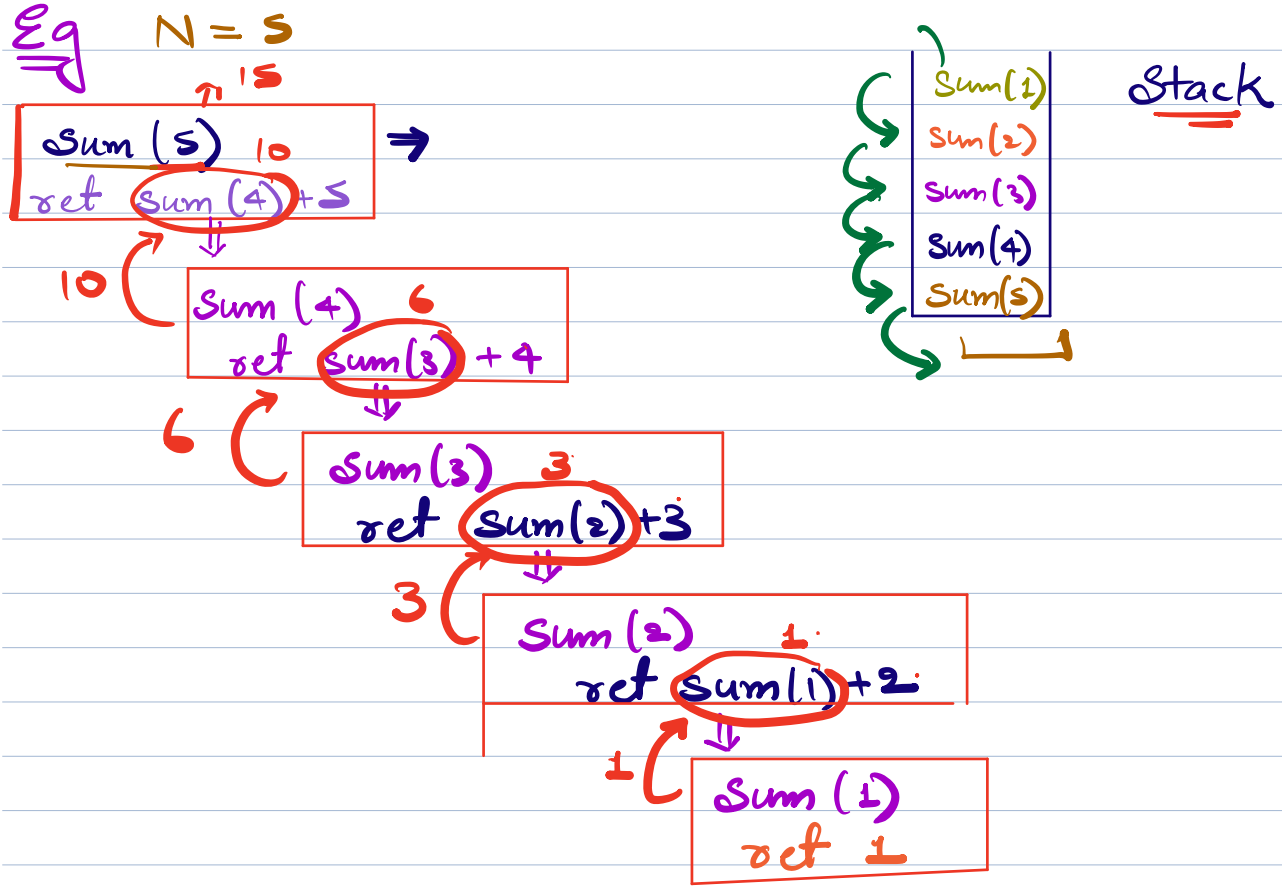
// Main Logic

return sum (N-1) + N;

}

N = 7

Eg N = 5



2) Factorial (N) using recursion

$3! = 3 \times 2 \times 1$

$5! = 5 \times 4 \times 3 \times 2 \times 1$

$\Rightarrow N! = N \times (N-1) \times (N-2) \times (N-3) \ldots\ldots 3 \times 2 \times 1$

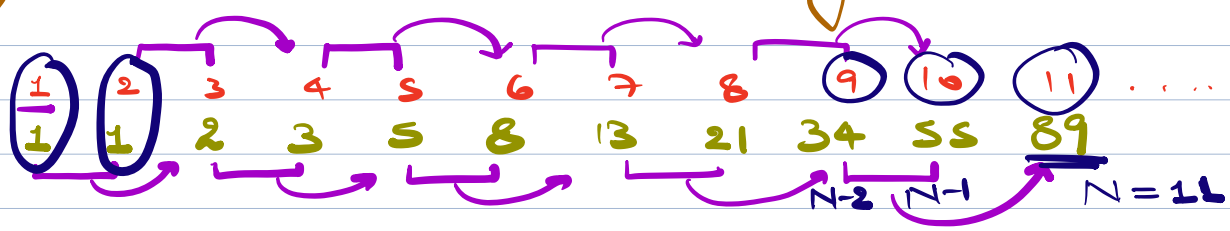
fact (N-1)

$1! = 1$

$0! = 1$

```
int fact (N) {
    // Assumption : fact(N) will return the
                    correct value of N!
    // Base Case
        if (N==1 || N==0) { return 1; }
    // Logic
        return  N * fact (N-1);
}
```

3) Fibonacci Series using Recursion




```
int fib(N) {
    // Assumption : fib(N) returns the correct
                    Nth fibonacci no.
```

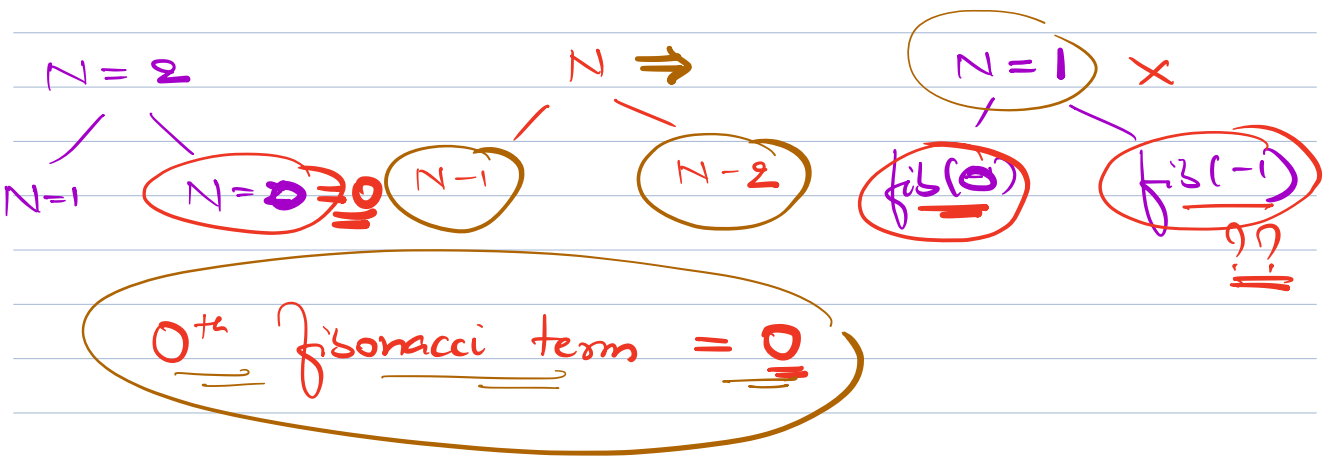
// Base Case

⇒ if (N==1 || N==2) { return 1; }

// Main Logic

fib(N) = fib(N-1) + fib(N-2);

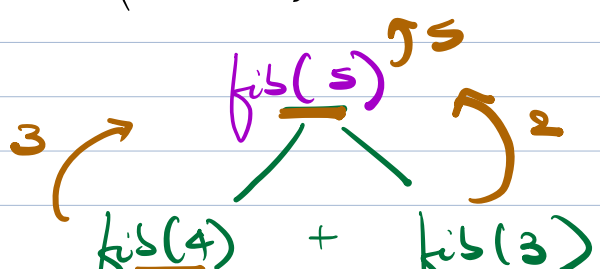
}



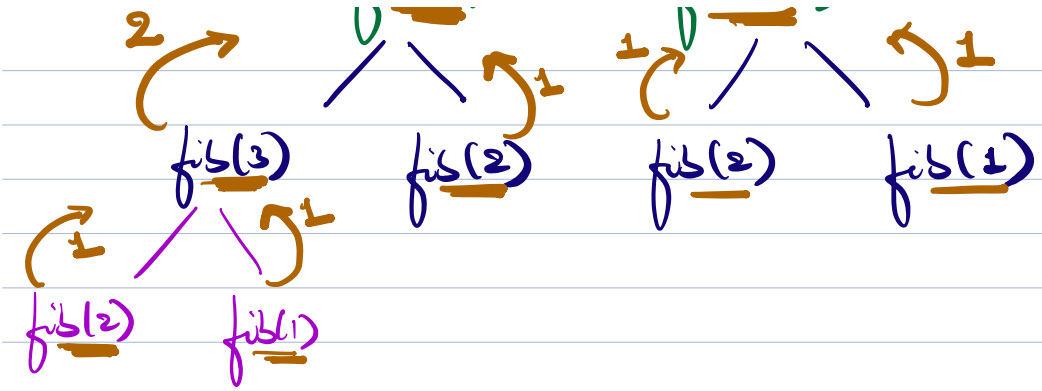
Base Case 2

⇒ [if (N==0) { return 0; }
if (N==1) { return 1; }] > if(N<=1){ return N; }

Dry Run (N=5)



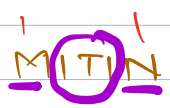
Recursion Tree

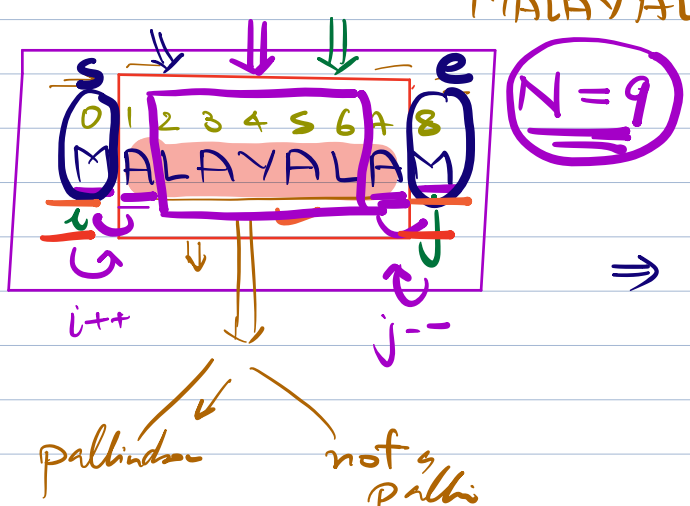


Given a string

Write a recursive program to check if the given string is a pallindrome.

Pallindrome ⇒ NAMAN
MADAM
NITIN
MOM
POOP
MALAYALAM





⇒ start, end

Code

boolean isPallindrome (str, s, e) {

0 ↑ s, N-1 ↑ e (initial call)

// Ass: isPallindrome( ) returns true if the str from s to e is a pallindrome

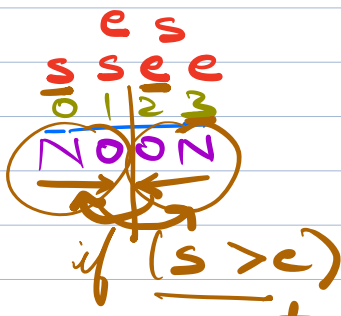
// Base Case

```
if (s >= e) { return True; }
```

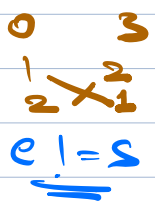
// Main Logic

```
if (str[s] == str[e]) {
    return isPallindrome(str, s+1, e-1);
}
else {
    return false;
}
```

s = e
ret True



if (s > e)
ret True



isPal ('NOON', 0, 3) {

⇓

isPal ('NOON', 1, 2)

isPal ('NOON', 2, 1)

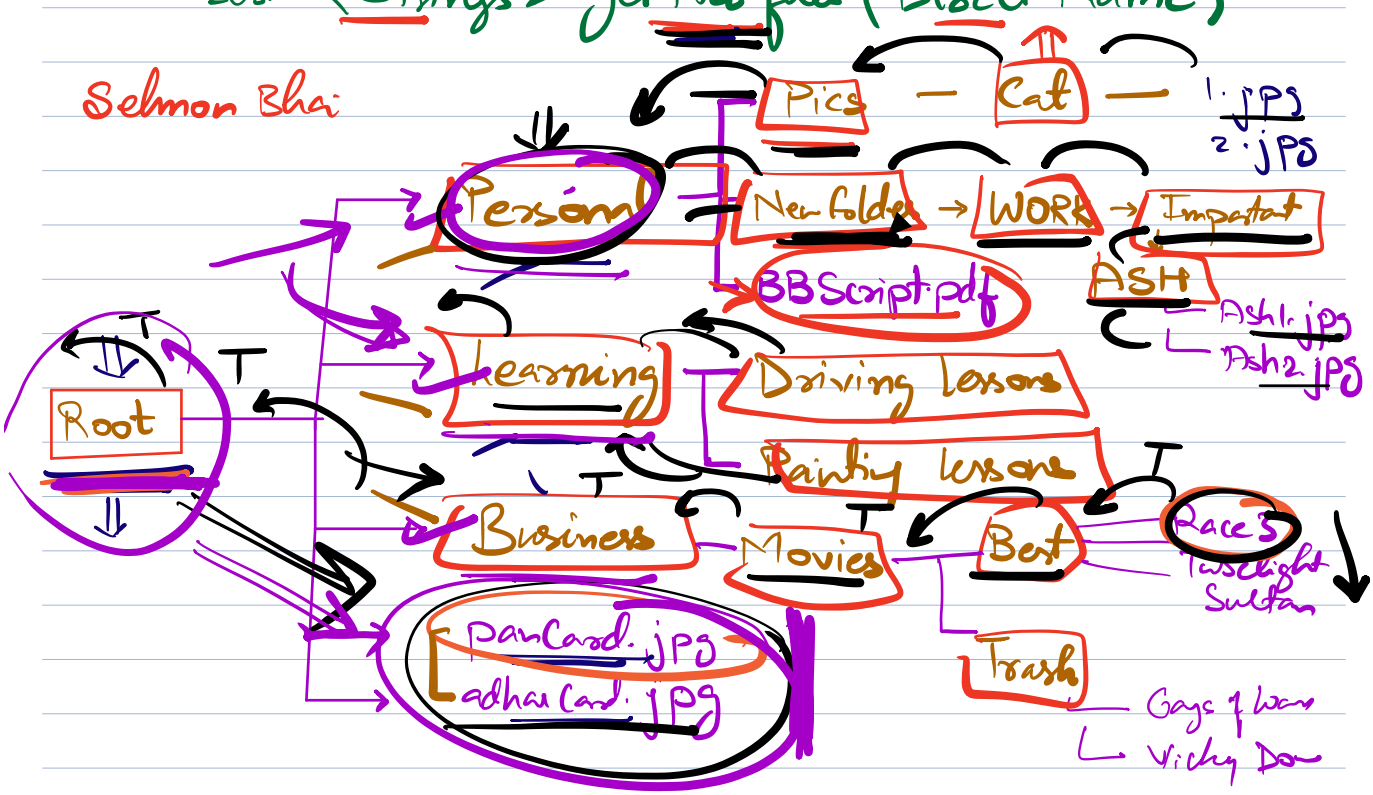
s > e

Facebook
Google

Given a dir structure.
Search a file in it

Given [List <String> getAllDirectories (DirectoryName)
List <Strings> getAllfiles (DirectName)

Selmon Bhai



Given a fileName ( ) return true if the file Exists anywhere in the directory.

Code

```
boolean search(DirectoryName, FileName){
// Ass: search(DN, FN) returns true if file
        is present anywhere in the directory.
    List<String> files = getAllFiles(DirectoryName);
// for(i=0; i<files.size(); i++){
        if(files[i].equals(fileName)){
            return True;
        }
    }
    List<String> directories = getAllDirectories(DN);
// for(i=0; i<directories.size(); i++){
    if(search(directories[i], fileName)){
        return True;
    }
}
return false;
}
```

Base Case

DirectoryName

TRUE

Eg

1) panCard.jpg

search (root, panCard.jpg) ⇒ { panCard.jpg, ash.jpg }

True

2) Race3.pdf

search (root, Race3.pdf) ⇒ <PC, AC>

⇓ { Personal, Learning, Business }

search (Personal, Race3.pdf)

⇓ { Cats, New folder }

search (Cats, Race3.pdf)

⋮

search (Learning, Race3.pdf).

⌐

Dosto

Sum(N) = 1+2+3+4. .... N-1+N

(2, N)

1 N

Sum(s, e) {

if (s == e) { return, e ; }

return s + Sum(s+1, e) ;

}