Observations :-

i) size keeps decreasing

ii) similar calls only size changes

iii) end call (last call) as stopping point

Use :- sorting algo :- merge sort, quick sort

DS :- Trees | BST | Heaps | Tries | Segment Trees

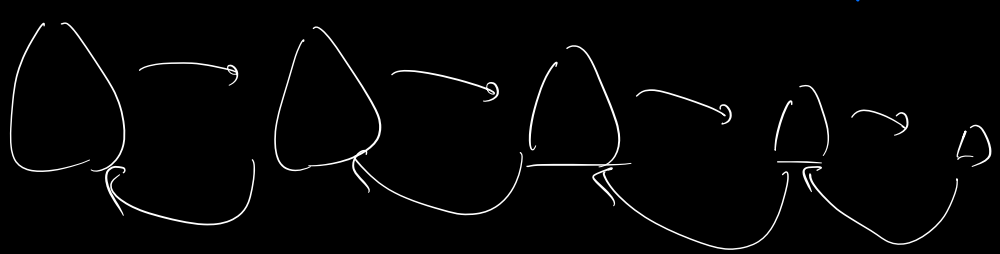
DP :- Dynamic Programming

Backtracking

Recursion :- function calling itself

technique of solving problem using smaller instances of the same problem

sub problem



SUM(N) = 1 + 2 + 3 + 4 + ——— + N-1 + N

SUM(N-1)

SUM(N) = SUM(N-1) + N

SUM(N-1) = 1 + 2 + 3 + 4 + ——— + (N-2) + (N-1)

SUM(N-1) = SUM(N-2) + (N-1)

SUM(1) = 1

- How to write recursive code
- How it works | dry-run
- TC/SC

How to write recursive code :-

ex ⇒ Find sum of N natural nos :-

Step I ⇒ assumption
decide what your func does

Step II ⇒ main logic
solve the main problem using sub problem

Step III → base condn

when should recursion stop

```
int sum(N) { // assume: this returns sum of N natural nos.
    // base condn
    if (N == 1)
        return 1
    // main logic
    return sum(N-1) + N
}
```

find factorial of N using recursion:

$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120.$

$4! = 4 \times 3 \times 2 \times 1 = 24.$

$5! = 5 \times 4!$

$0! = 1$

$1! = 1$

$N! = N \times (N-1)!$

```
int fact(N) { // assume => fact(N) returns factorial of N
// base condn
  if( N==0 || N==1)
     return 1
// main logic
     return N * fact(N-1);
}
```

#3 Find Nth fibonacci number

N =>	0	1	2	3	4	5	6	7	8	9	...
	0	1	1	2	3	5	8	13	21	34	...

fib(8) = 21

= 8 + 13

= fib(6) + fib(7)

=> fib(8) = fib(7) + fib(6)

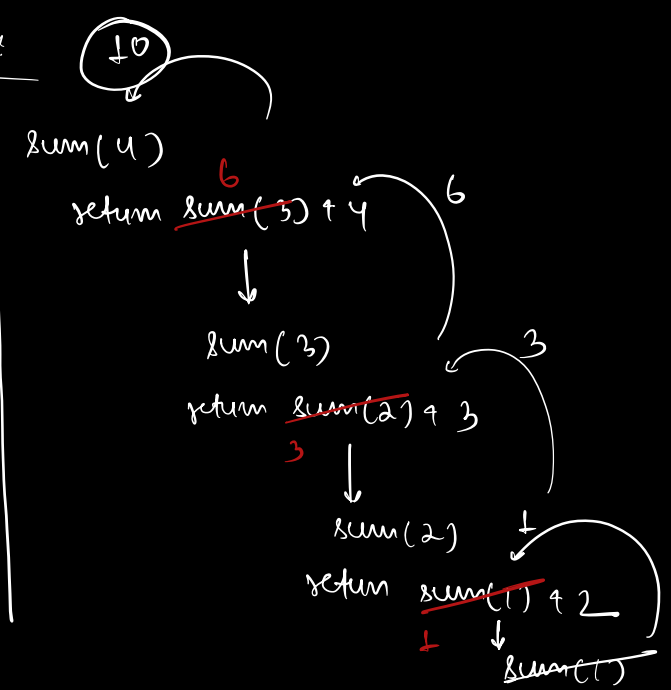
=> **fib(N) = fib(N-1) + fib(N-2)**

```
int fib(N) { // Assume: fib(N) returns fibonacci no. for N
    // base condn
    if(N==0)
        return 0
    else if(N==1)
        return 1
    // main logic
    return fib(N-1) + fib(N-2);
}
```

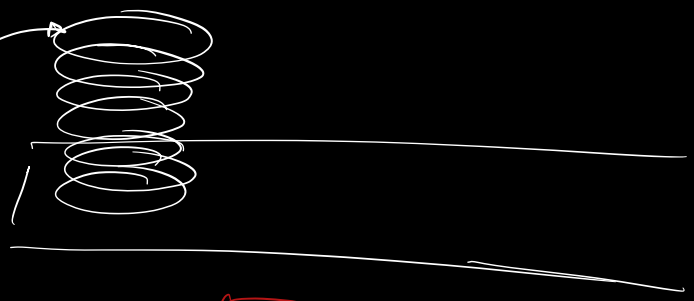
=> Working of recursion :-

Sum of N natural nos:

```
int sum(N) {
    if(N==1)
        return 1
    return sum(N-1) + N
}
```



Call Stack

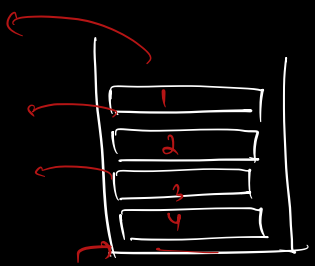


Stack

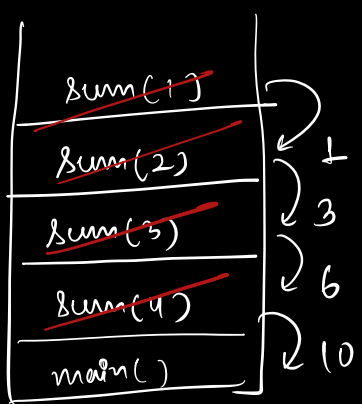
↓

LIFO

Last In First Out



Call Stack → Stack that contains all function calls.

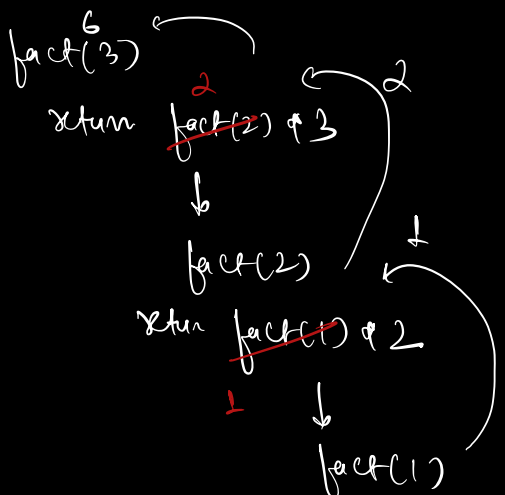


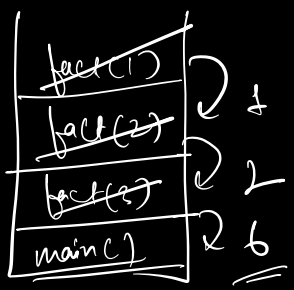
Call Stack

```
int sum(N) {
    if(N==1)
        return 1
    return sum(N-1) + N
}
```

factorial of N

```
Int fact(N) {
    if(N==0 || N==1)
        return 1
    return fact(N-1) * N
}
```

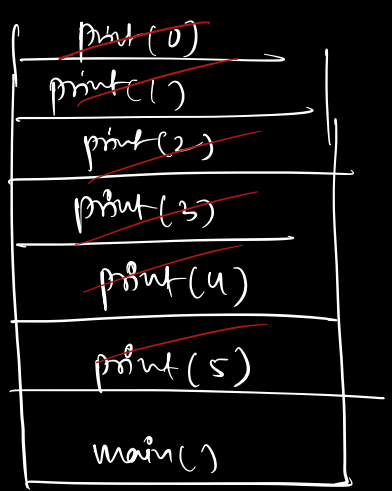




call stack

Q 1. Given a no. N, print all the no. from N to 1 in decreasing order, using recursion

N => 5 4 3 2 1




```
void print(N) {
    if (N==0)
        return;

    sout(N)
    print(N-1)
}
```

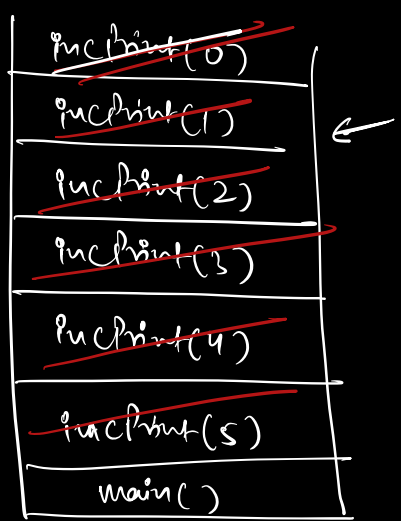
doing work while going deep

5 4 3 2 1

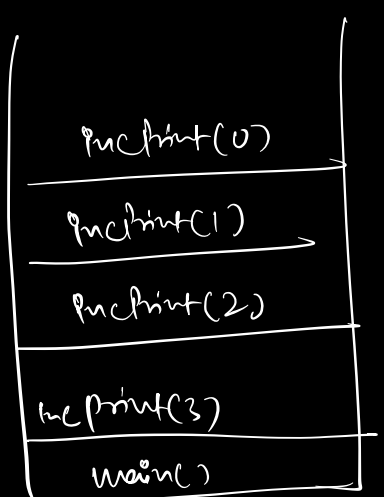
Q 2. Given a no. N, print all the no. from 1 to N in increasing order, using recursion

eg: N = 5 → 1 2 3 4 5

```
void incPrint(N){
    if(N==0)
        return;
    incPrint(N-1)
    sout(N) ←
}
```



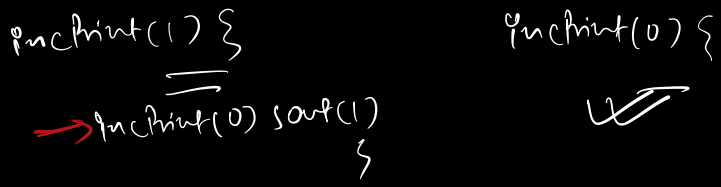
① ② ③ ④ ⑤




```
void incPrint(N){
    if(N==0)
        return;
    incPrint(N-1)
    sout(N) ←
}
```

```
incPrint(3){
    incPrint(2)
    sout(3)
}
```

```
incPrint(2){
    incPrint(1)
    sout(2)
```

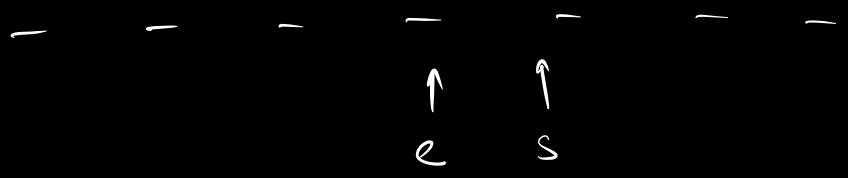


Q 3. Given a string, write a recursive code to check if it is a palindrome

palindrome ⇒ W O W

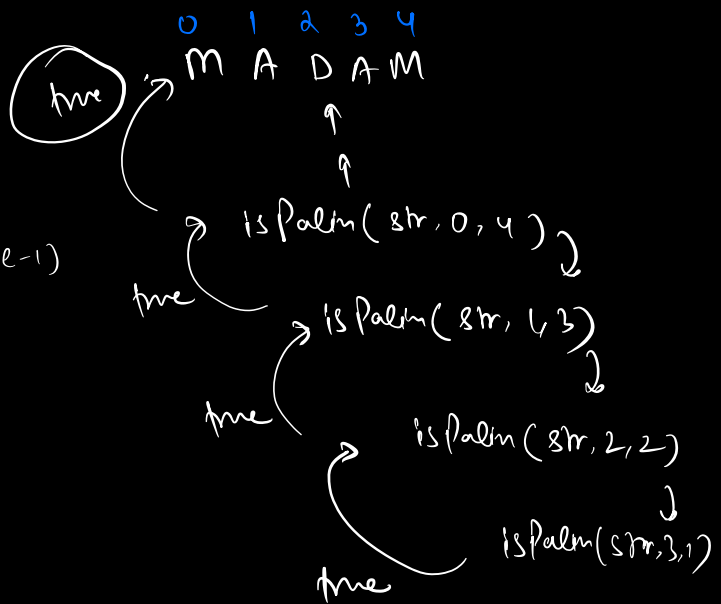
M A D A M

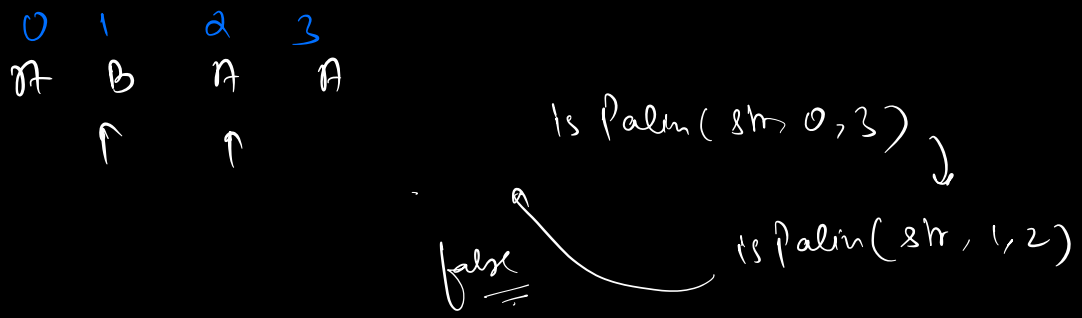
L E V E L




```
boolean isPalin( str, s, e) {
    if (s>e)
        return true
    if( str[s] == str[e] {
        return isPalin( str, s+1, e-1)
    }
    else {
        return false
    }
}
```

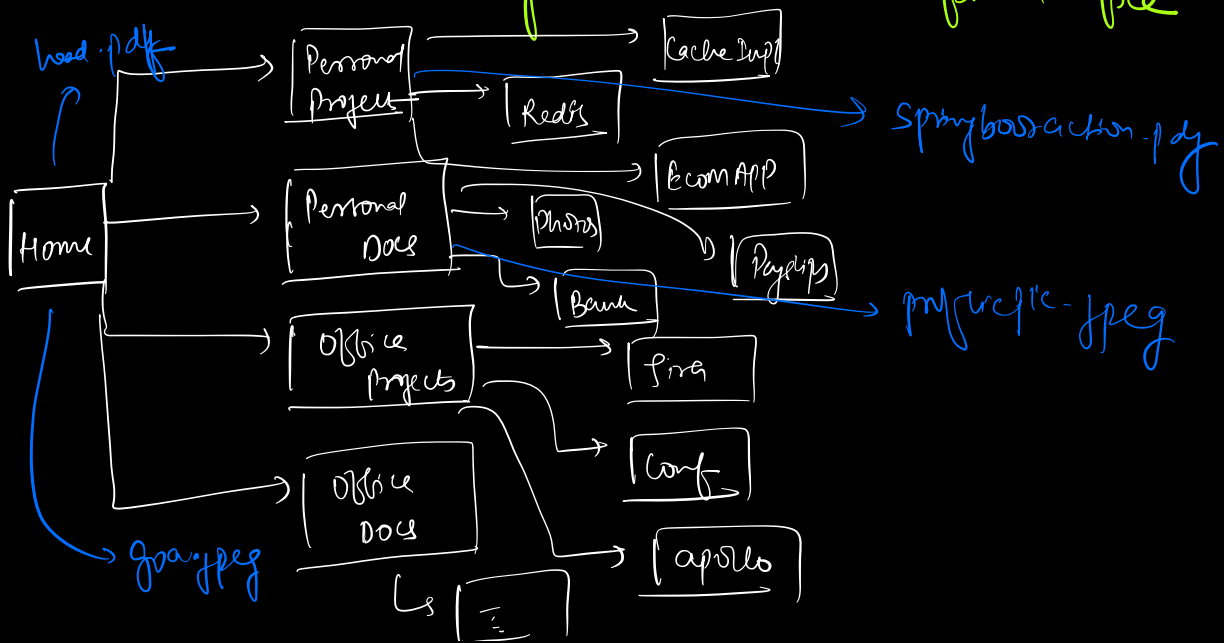
```
boolean isPalin( str, s, e) {
    if(s>e)
        return true
    if( str[s] == str[e] {
        return isPalin( str, s+1, e-1)
    }
    else {
        return false
    }
}
```

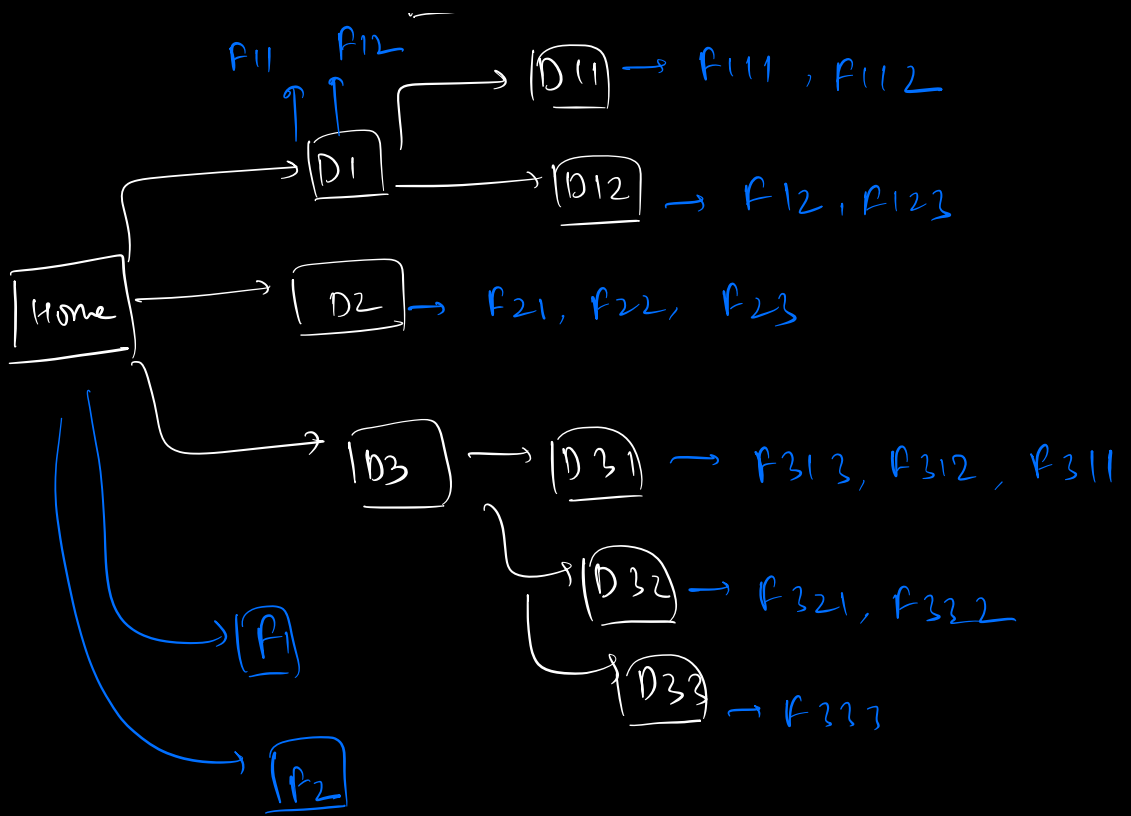




Facebook | Google | Amazon

Q 4. Given a directory structure, search for a file





given

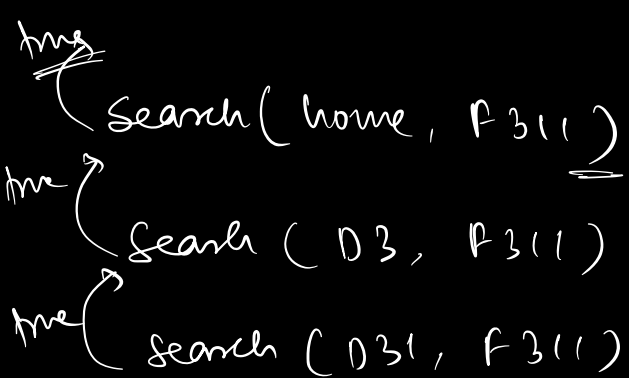
```
List<String> files = getAllFiles( directoryName)

List<String> directories = getAllDir( directoryName)
```

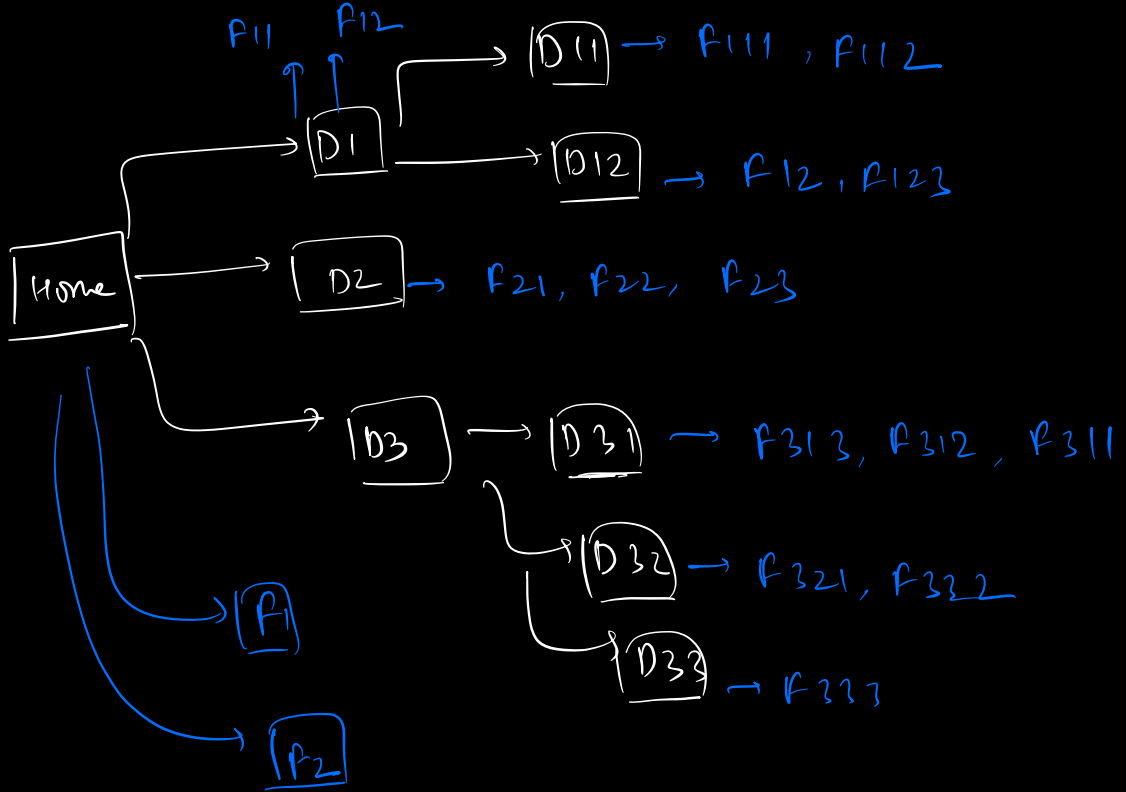
```
boolean search( HOME , F311 )  dirName , fileName ) {
    List<String> files = getAllFiles( dirName );
    for( i=0; i< files.size(); i++ ) {
        if( files[i] == fileName ) {
            return true
        }
    }
    List<String> dir = getAllDir( dirName ) {
    for( i=0; i< dir.size(); i++ ) {
        if( search ( dir(i), fileName )) {
            return true
        }
    }
    return false
}
```

base condn

D1 D2 D3

D31 D32 D33




f11
f12
D11
f111 , f112
D1
D12
f12 , f123
Home
D2
f21 , f22 , f23
D3
D31
f313 , f312 , f311
D32
f321 , f322
D33
f333
f1
f2