2 classes on Recursion

Recursion :

Tod
- → How to write recursive code
- → Working
- → TC/SC

Next Session

Why?
- → Merge sort
- → Binary Trees
- → Dynamic Programming
- → Backtracking

Recursion : { function calling itself }

Solve a problem using smaller version of the same problem.

↳ subproblem

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

1 + 2 + 3 + 4 + - - - + N-1 + N

sum(N-1) + N

Recursion Code

1) Assumption : Decide what your function does, and assume it does exactly that

2) Main Logic : Solving Assumption with subproblem

3) Base Condition : When should code stop.

Smallest value for which we know answer.

```
int sum(N) {        Calc sum of first N natural no.s.
Assumption:   return sum of first N
                   natural nos.
Base Case:     if ( n==1)
                    return 1
Main logic:   return  ( sum(n-1) + n)

}
```

```
int fact(N) {        Calculate factorial of N
                      1×2×3×4× ....× N
Assumption:  return factorial value of N

Base case:     if ( n==1 )
                    return 1
Main Logic:   return ( fact(n-1) * n   )

}
```

$$f(N) = \underbrace{1\times 2\times 3 \times \ldots\ldots N-1}_{f(n-1)} \times N$$

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

	1	2	3	4	5	6	7	8	9	10	11	12
fib:	1	1	2	3	5	8	13	21	34	55	89	144

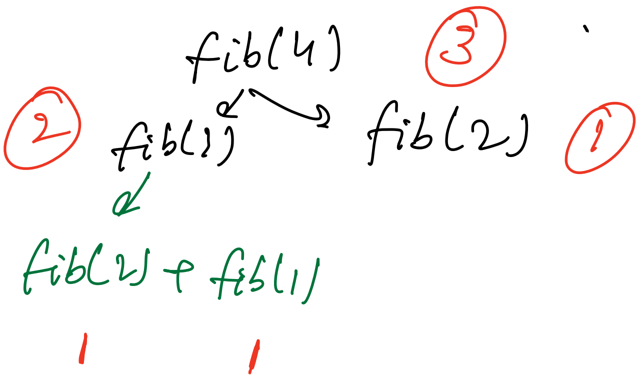
int fib(N) { N^{th} fibonacci number

Assumption: Return n^{th} fib number

Base Case : if(n==1 || n==2)
return 1

Main logic return (fib(n-1) + fib(n-2))

}

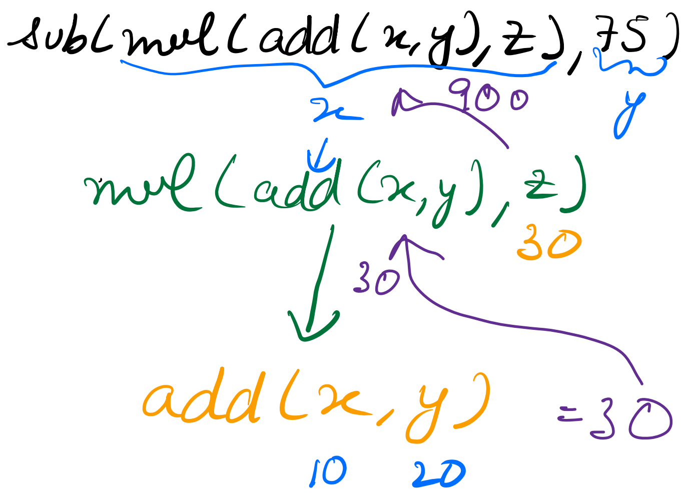


Example :

```
int add(x, y){
   return x+y
}

int mul(x, y){
  return x*y
}

int sub(x, y){
  return x-y
}
```

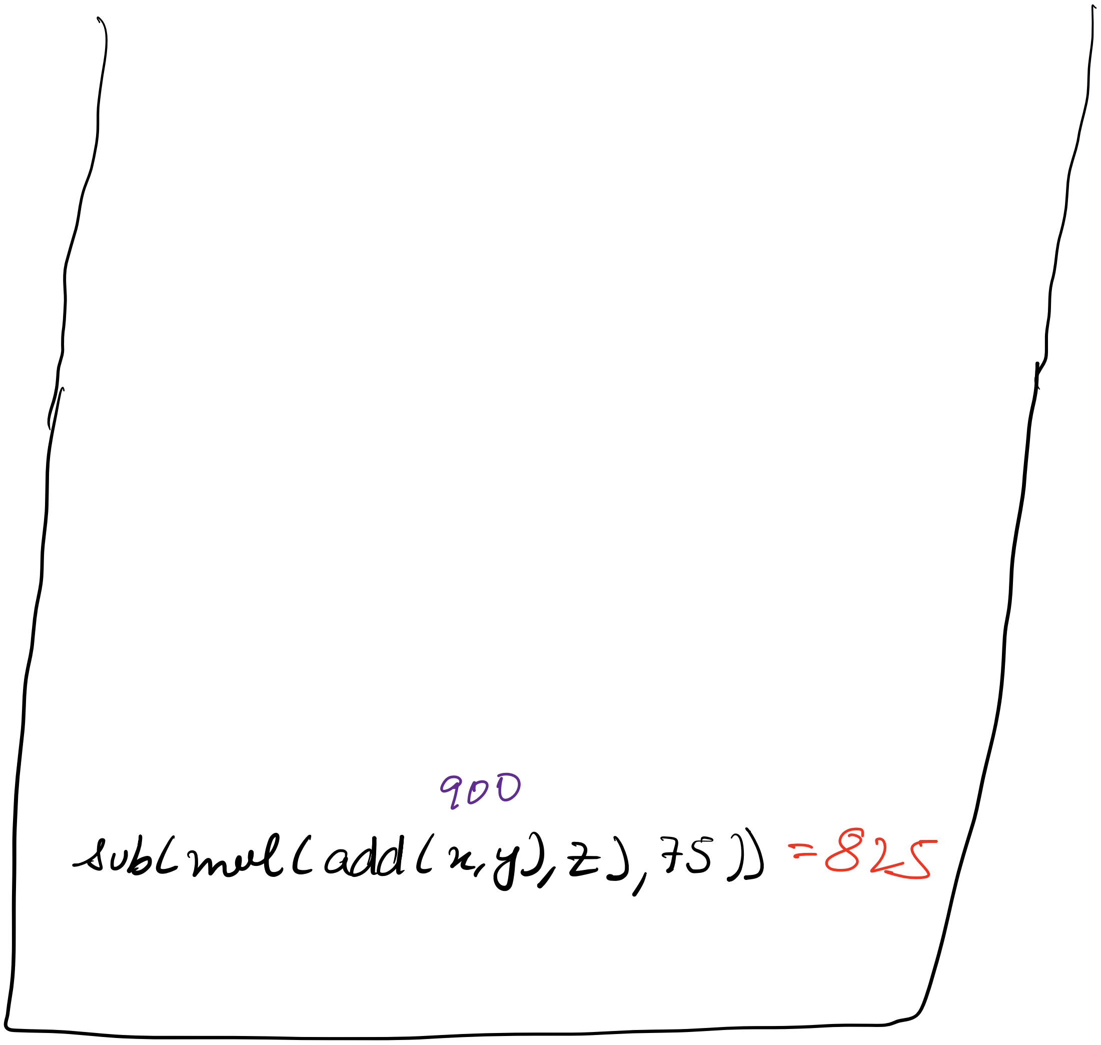
```
main(){
  int x=10, y=20, z=30
  int a= add(x,y)       a= 30
  int m= mul (a,z)      m= 900
  int s= sub(m,75)      s= 825
  print(s)              825
}
```

```
main(){
 int x=10, y=20, z=30
print(sub(mul(add(x,y),z),75))
}
```



825

Internal working (Call Stack)



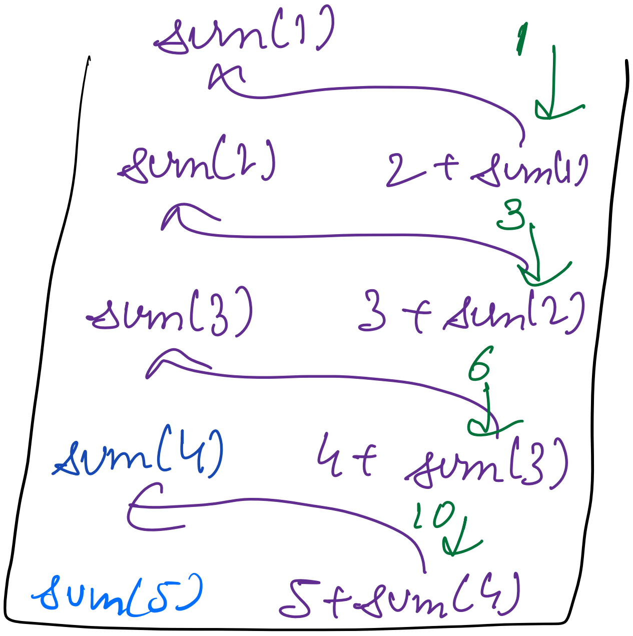
sub(mul(add(x,y),z),75)) =825

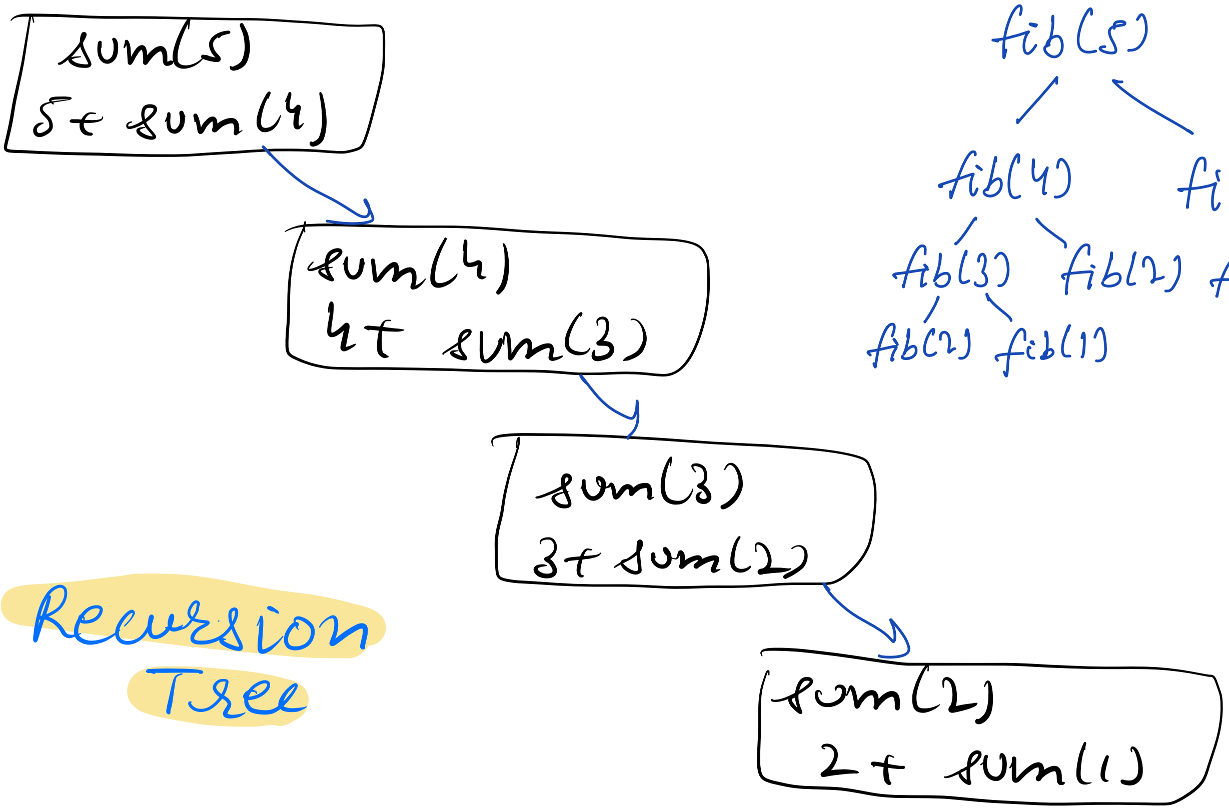
Works like an Idli cooker

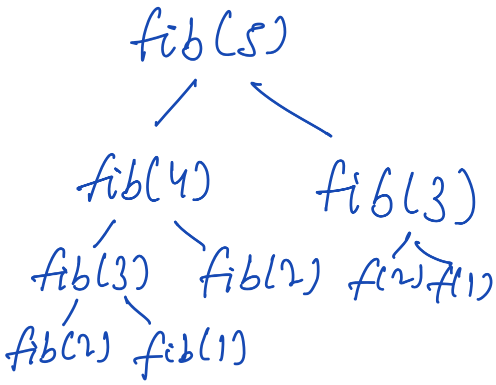
```
int sum (N) {
 if (N==1) return 1
 return   N+ sum (N-1)
}


main() {
  print (sum(5))
}
```

15

1+2+3+4+5 = 15





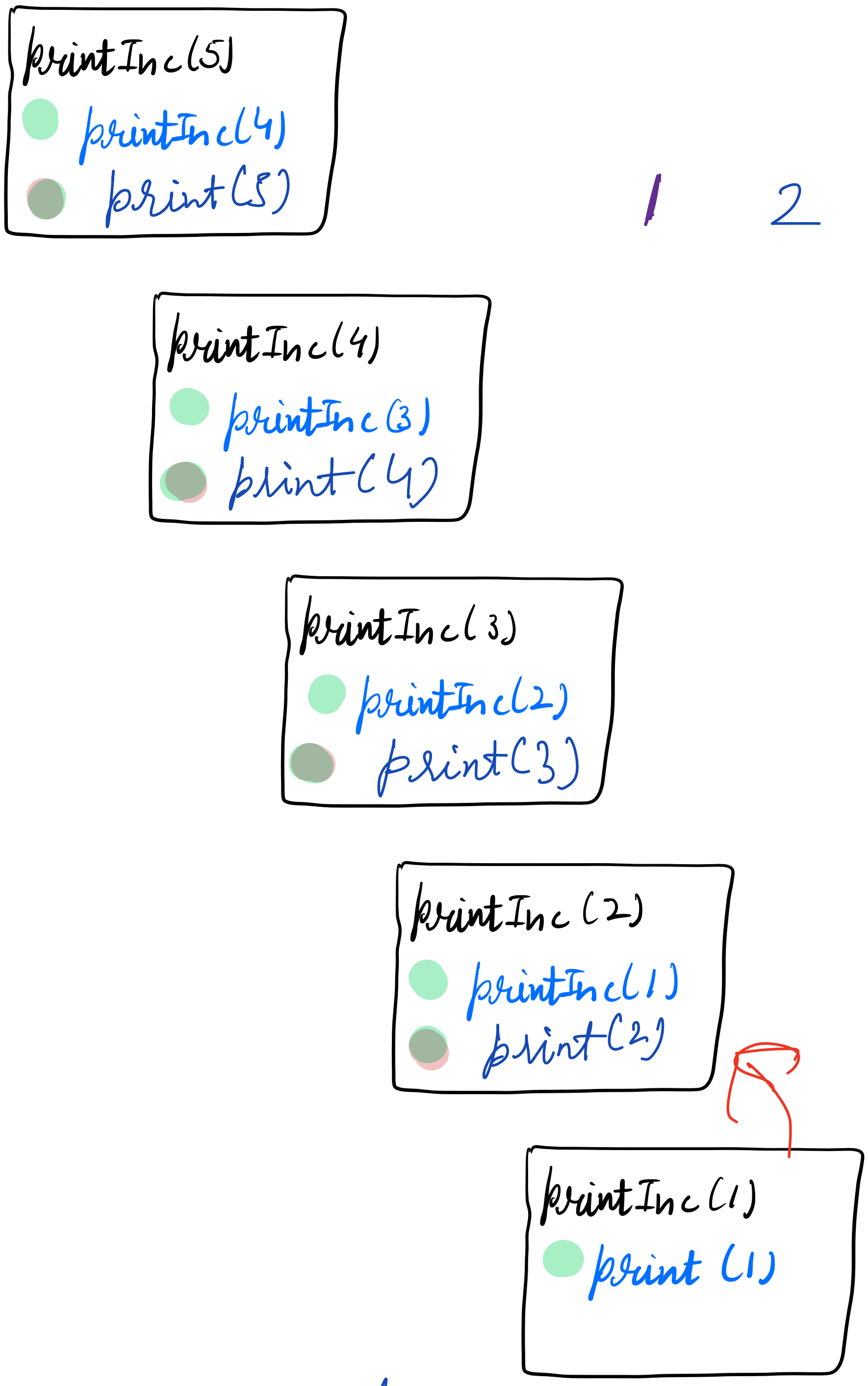


Recursion Tree

```
                         → print  1 2 3 4 ...... N
void printInc(N) {
    Assumption:      prints  1  to  N

    Base Case:       if (n==1)
                         print 1     return;
    Main Logic:      printInc(n-1)
                       print(N)

}
```

p(N) = 1 2 3 4 ....N-1 N

```
main() {
    printInc(5)
}
```



1 2 3 4 5

s=0 e=1

s=1 e=0

0 1

a a

e s

Q Given string check whether palindrome

Eg = abacaba ⇒ true
abcd ⇒ false.

Solve using recursion.

bool ispal(str, $\overset{\text{start}}{s}$, $\overset{\text{end}}{e}$) {

Assumption: Return if the substring str[s:e] is palindrome

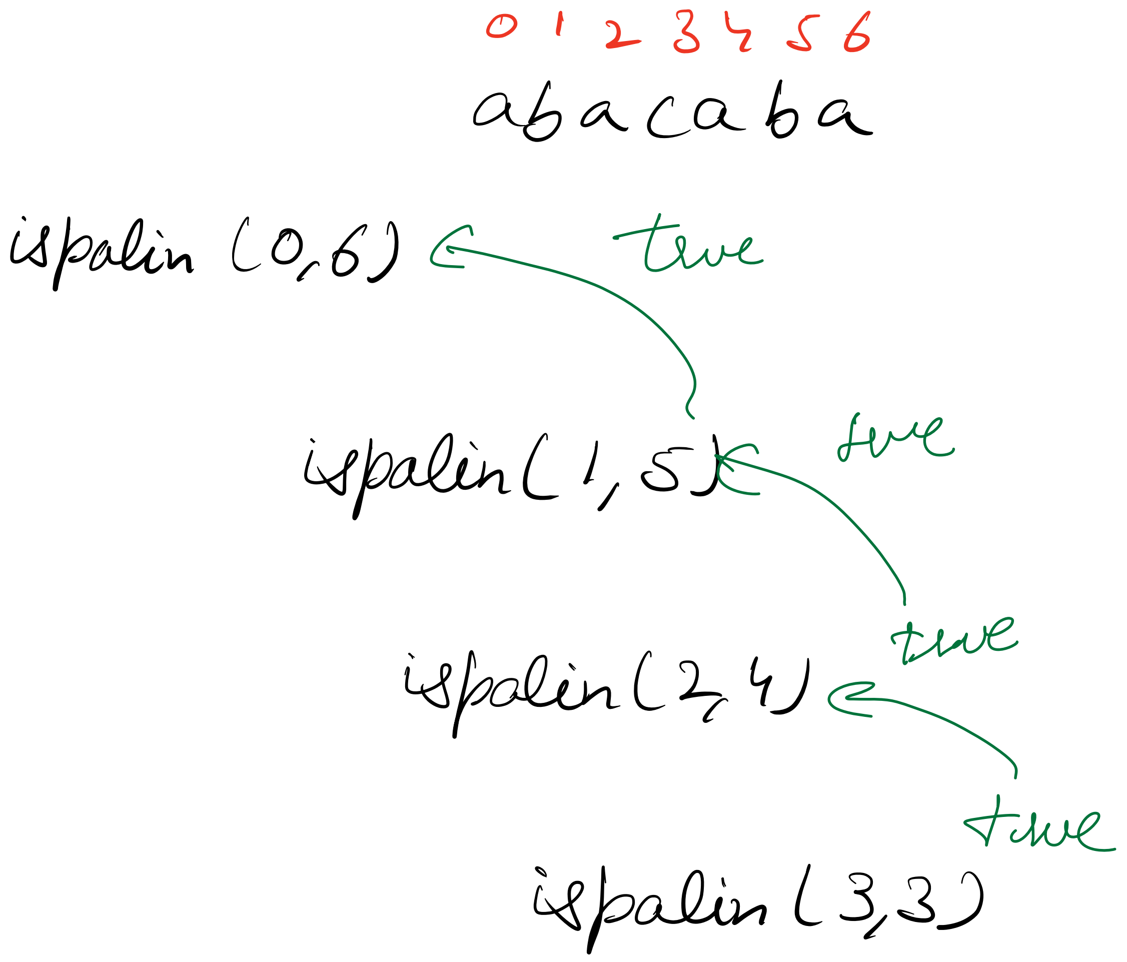
Base case: if (s >= e) return true

Main logic:

if (str[s] != str[e])
return false

else return ispalin(str, s+1, e-1)

}



0 1 2 3 4 5 6

a b a c a b a

ispalin (0,6) ← true

ispalin(1,5) ← true

ispalin(2,4) ← true

ispalin(3,3) true

0 1 2 3
a b c a

ispal(0,3) ← false

ispal(1,2) false

0 1 2 3
a c c a

ispal(0,3)

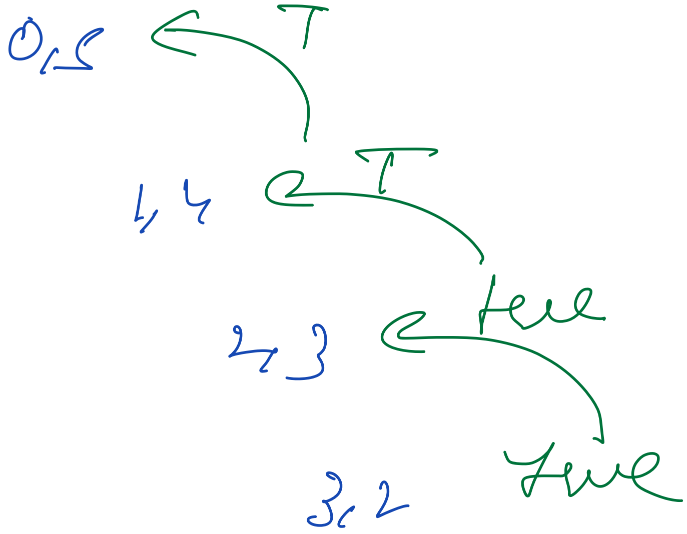
ispal(1,2)

ispal(2,1)

0 1 2 3 4 5

a d p p d a

{done}



M — O — T

DP