Recursion - 1

- How do we write
- How does it work
- How to cal T.C/S.C (later)

Adv.
- Merge / Quick sort
- Trees / BST / Heaps / Tries / Seg. Trees
- Dynamic programming
- Backtracking

Observations:

① Size keeps decreasing

② Dolls are similar

③ End doll (last doll)

Recursion ⇒ **Function calling itself**

Solving a problem using smaller instances of same problem

⇓

subproblem.

$$\text{sum}(N) : \underbrace{1+2+3+\cdots+(N-1)}_{\text{sum}(N-1)} + N$$

$$\text{sum}(N) : \underbrace{\text{sum}(N-1)}_{\text{subproblem}} + N$$

Steps

① Assumption

↳ Decides what the function does?

```
int sum(N){
//Assumption sum(N) gives
sum of N natural nos.
//Base condion
 if (N==1)
```

② **Main logic**

Solves the bigger problem using subproblem

③ **Base Condition**

When recursion should stop.

```
        return 1;

  //Main logic
  return sum(N-1) + N;
  }
```

#Factorial of N

$$N! = \underbrace{1 * 2 * 3 * 4 * \cdots * (N-1)}_{(N-1)!} * N$$

$$1! = 1$$

$$0! = 1$$

① Assumption ⇒ fact(N) gives N!

② Main logic ⇒ fact(N) = fact(N-1) * N

```
int fact (N){
//Base condition
if (N==0)
        return 1;

//Main logic
return fact (N-1) * N;
}
```

$$fact(1) = \underset{\overset{\downarrow}{1}}{fact(0)} * 1$$

$$fact(0) = \cancel{fact(-1)} * 0$$

Fibonacci Series

Golden Ratio

N:	0	1	2	3	4	5	6	7	8	9	.....
	1	1	2	3	5	8	13	21	34	55	.....

31st fibonacci no?

```
int fib(N){
  //Base Condition.
  if (N==0 || N==1)
      return 1;

  //Main logic
  return fib(N-1) + fib(N-2)

}
```

① Assumption

Fib(N) gives Nth fibonacci No.

② Main logic

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

③ Base Condition

$$fib(\underline{0}) = \underline{fib(-1)} + \underline{fib(-2)}$$

$$fib(\underline{1}) = fib(0) + fib(-1)$$

$$fib(\ ) = fib(1) + fib(0)$$

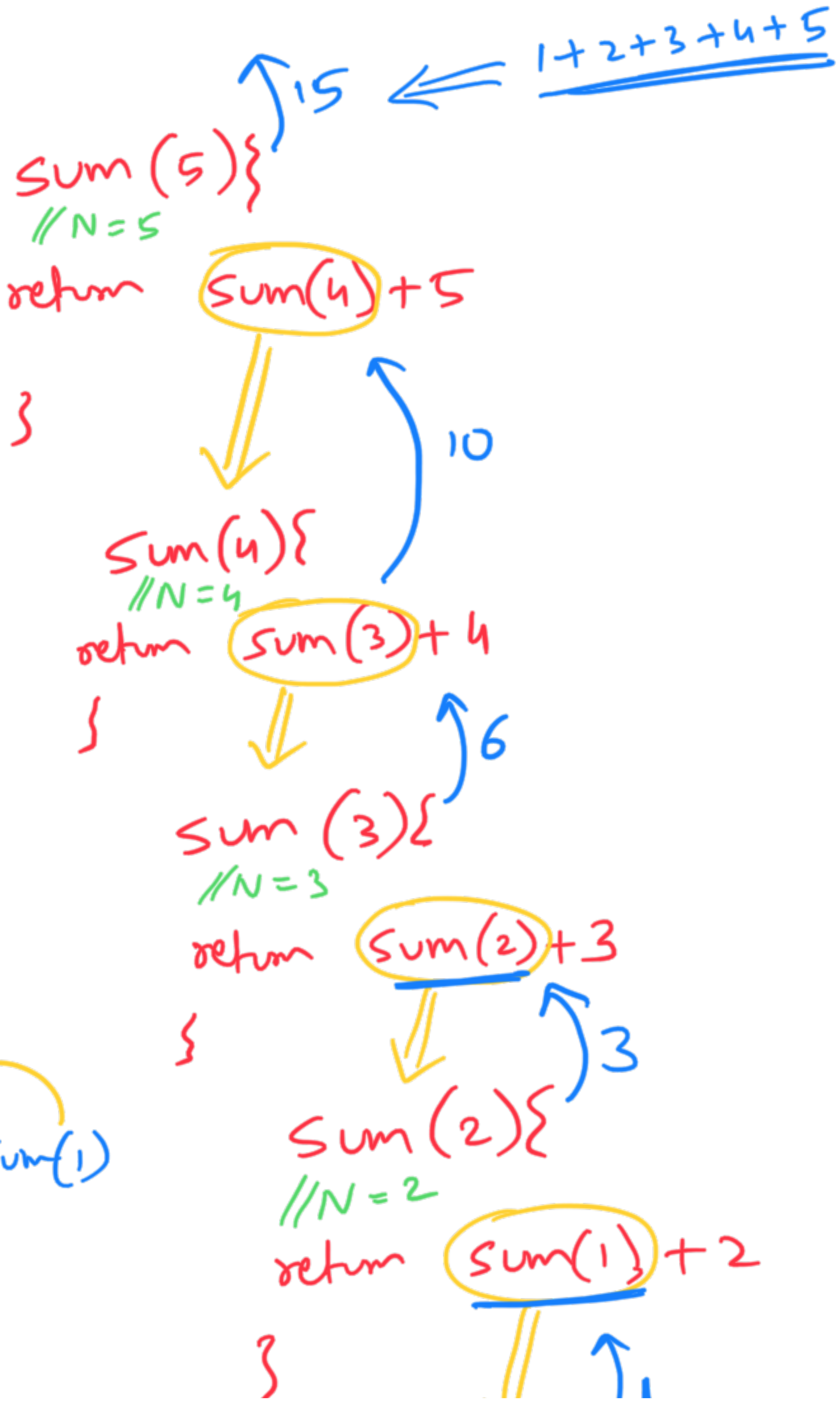
fib(2) = fib(1) + fib(0)

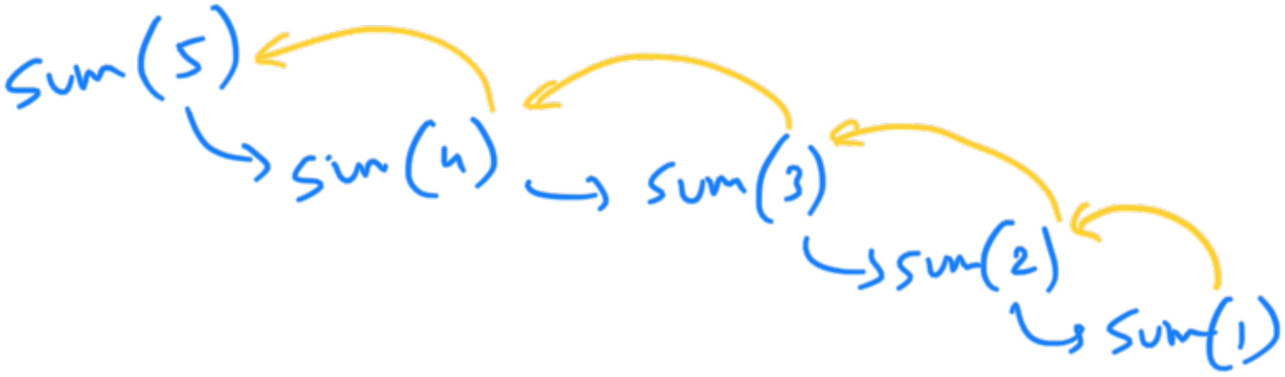
Sum of N natural nos

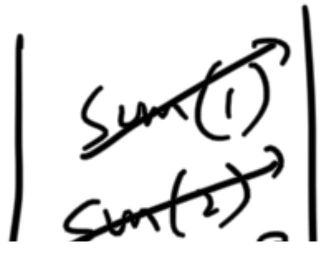
```
int sum(N){
if (N==1)
        return 1;

return sum(N-1)+N;

}
```





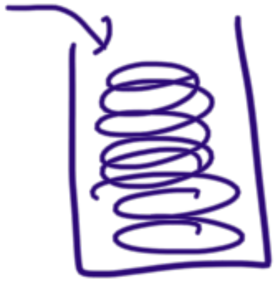


LIFO → out

~~Sum(3)~~
~~Sum(4)~~
~~Sum(5)~~

Last In first

Stack.



```
sum(1){
  //N=1
  X
}
```

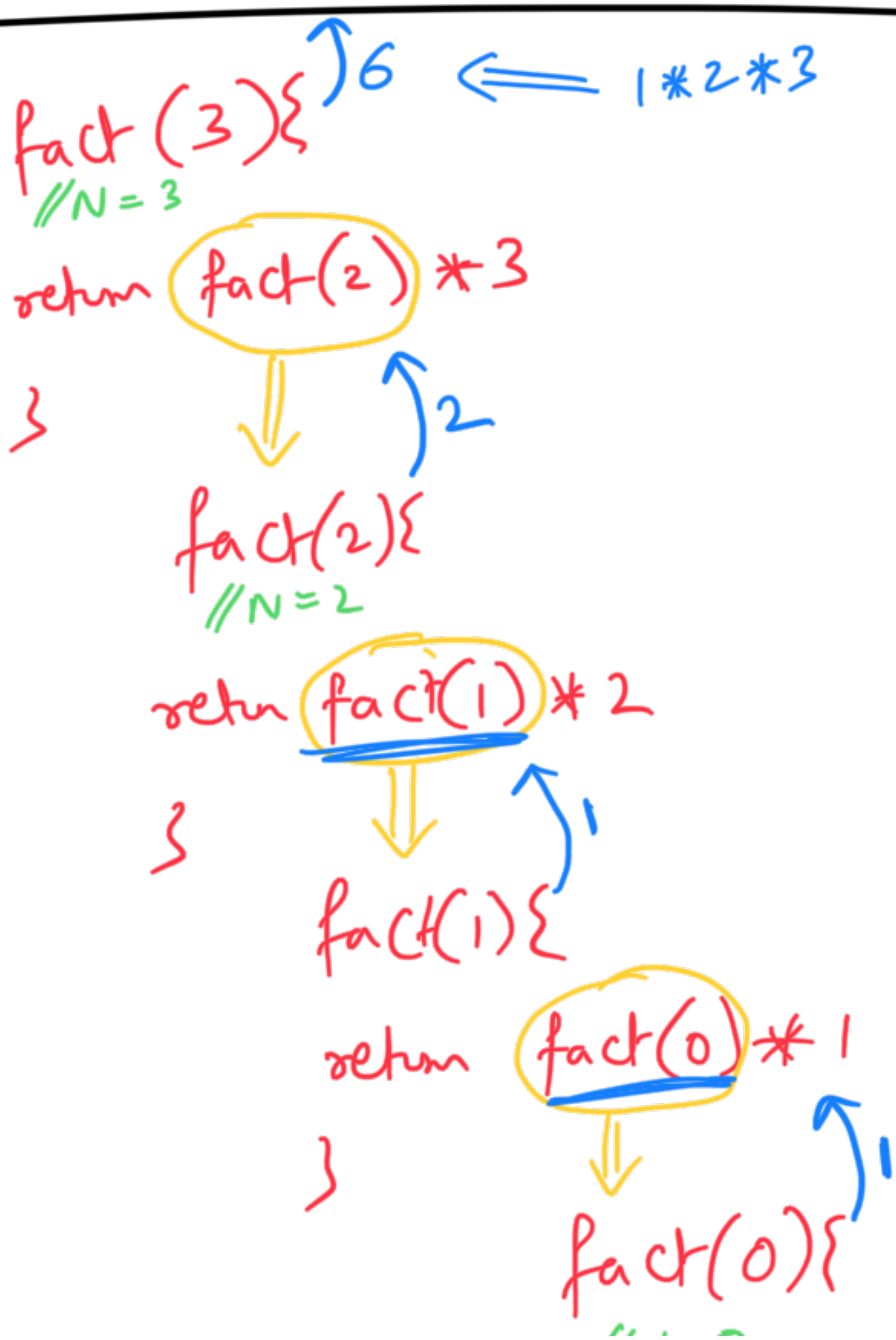
factorial of N.

```
int fact (N){
  if (N==0)
        return 1.

return fact(N-1)*N

}
```

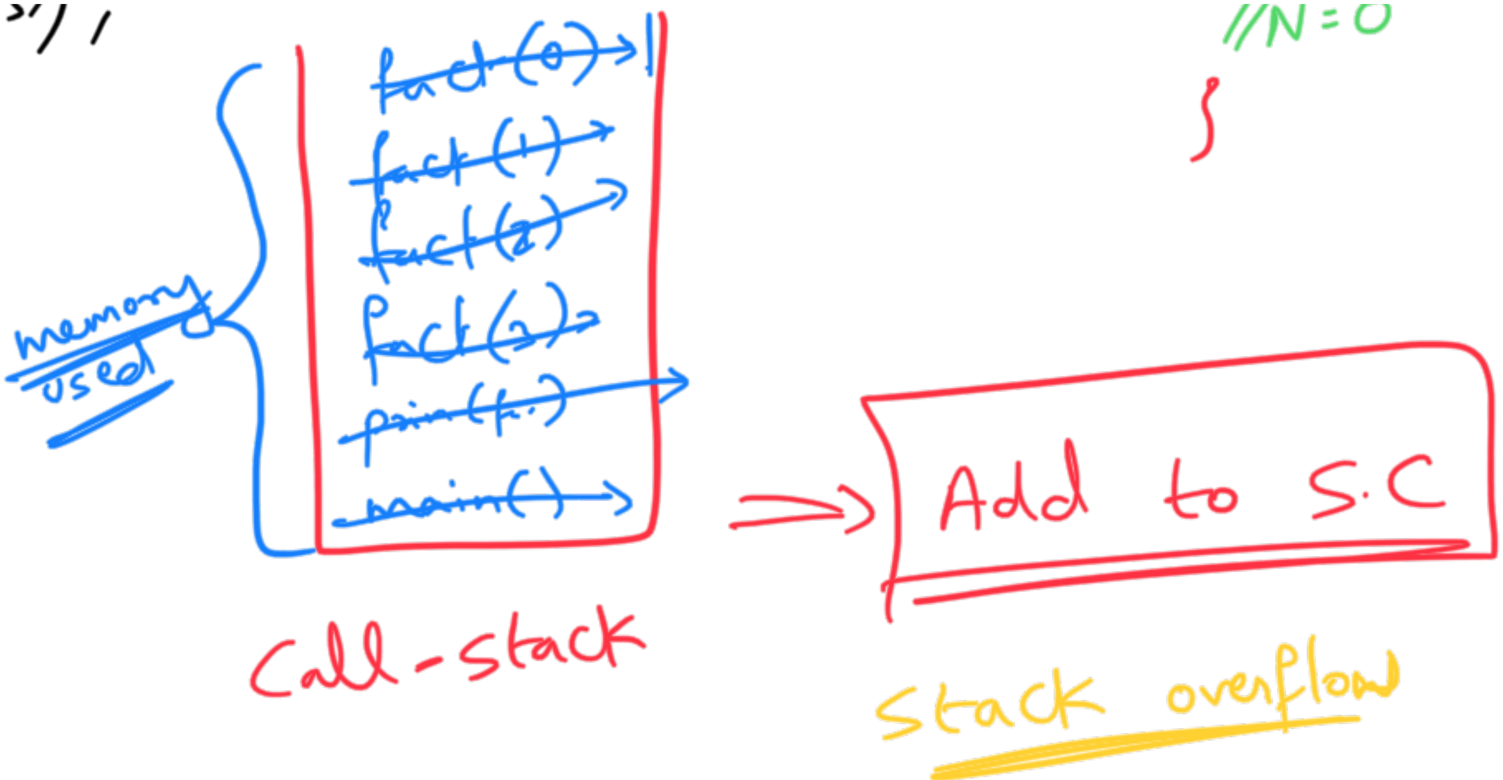
```
int main()
{
     (fact(3)).
```



```
fact (3){
//N=3
return fact(2) *3
}
      fact(2){
      //N=2
      return fact(1) * 2
      }
            fact(1){
            return fact(0) * 1
            }
                  fact(0){
```

```
    print( fact(3));
  return 0;
}
```

```
//N=0
}
```



Q. Given a no. 'N'.
Print all the nos. from 1 to N in increasing order using recursion.

```
void func(N){
    if(N==0)
        return;
    func(N-1)  ——→ ②
    print(N)   ——→ ③
```

print →

func(N) = 1 , 2 , 3 , 4 , ... , N-1 , N

main logic

func(N)

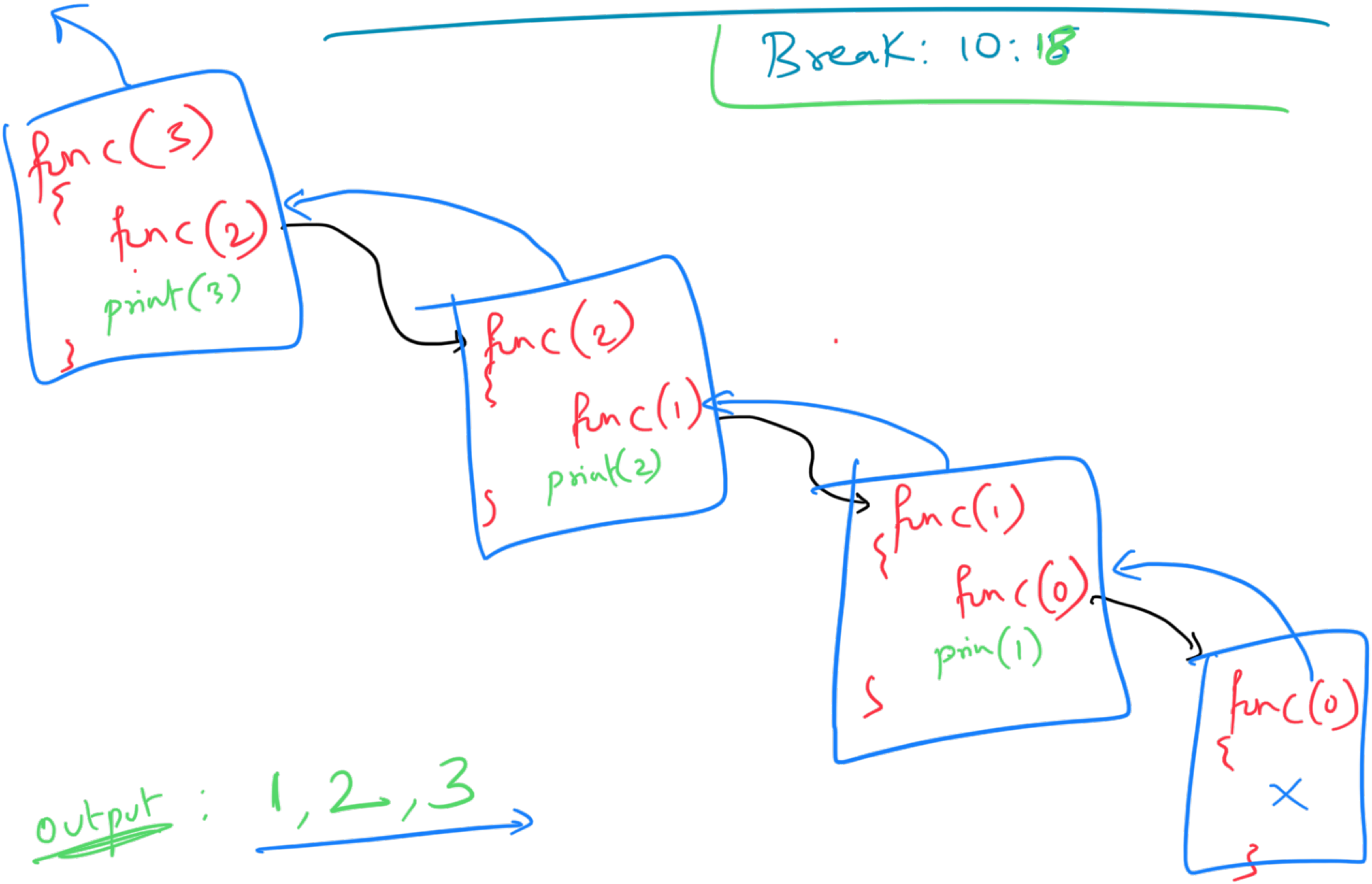
= 1 , 2 , 3 , 4 , . . . N-1 | N

print(N)

}

func(N-1)

Break: 10:18



output : 1, 2, 3

Q Print in decreasing order. fun(N)

Q.

$\Rightarrow N, \underbrace{N-1, \ldots, 3, 2, 1}_{\text{fun((N-1)}}$

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

    print(N);
    func(N-1);


}
```

Q. Given a string.
Check if string is a palindrome
using recursion.

Palindrome : → $\boxed{\overset{s}{r}aceca\overset{e}{r}}$ ←
madam
mom
nitin
तेजू के पापा के जूते
e

s

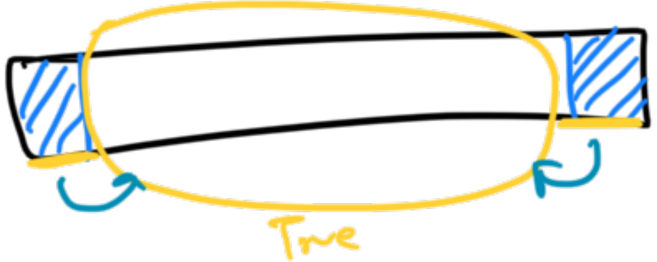
```
bool isPal(str, s, e){
  //Base Case
  if (s >= e)
          return true

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

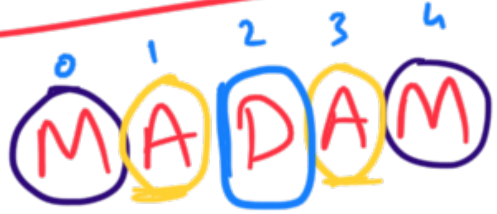
① Assumption

isPal(str, s, e)

↳ if str from 's' to 'e' is palindrom



Dry Run



isPal(str, 0, 4)

isPal(str, 1, 3)

true

isPal(str, 2, 2)

True

Cschook

Given a dir structure. Search for a file in it.

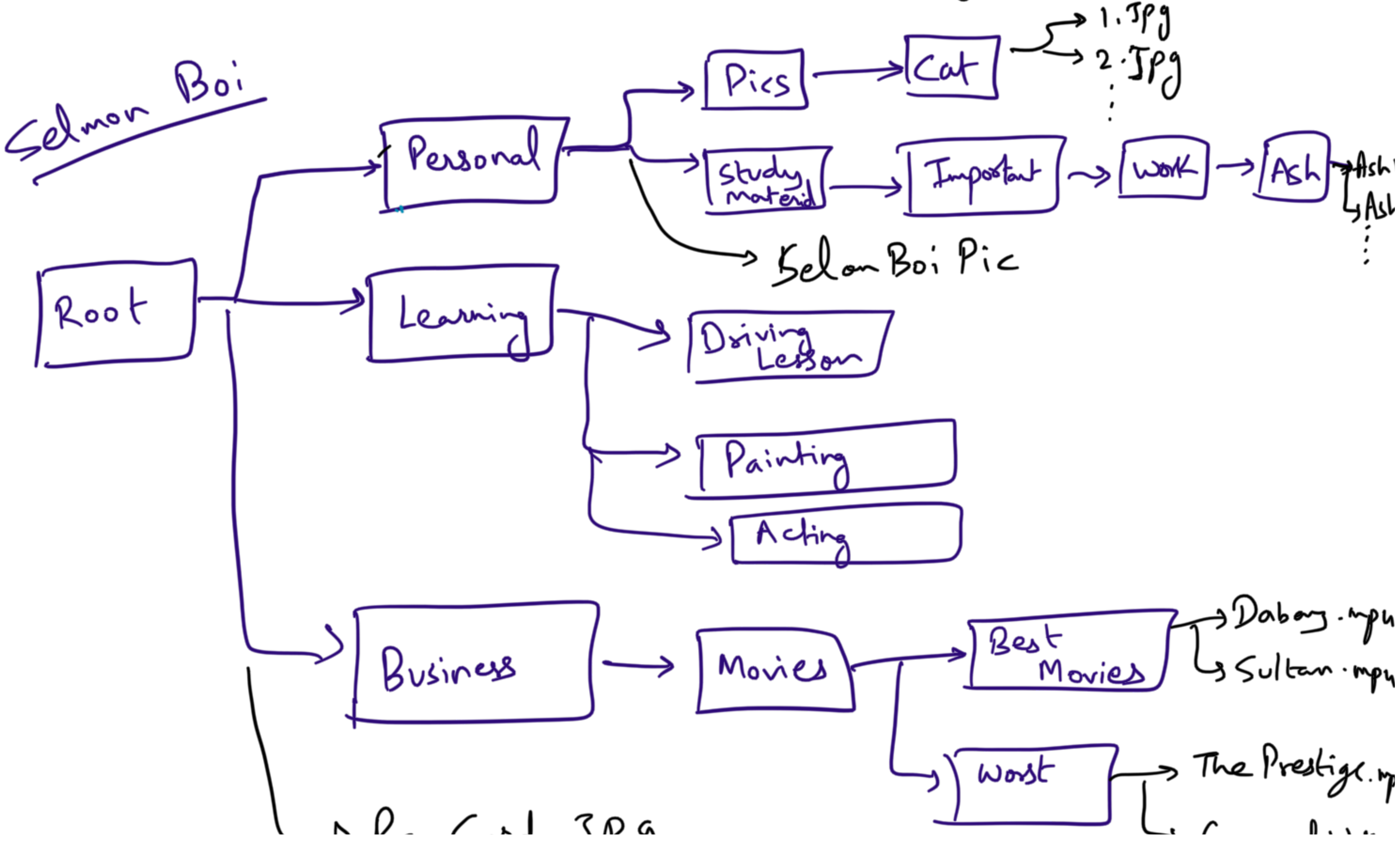
Facebo
Google

Q. Given a

Given:

List<str> getAllDirectories(Directory Name)

List<str> getAllFiles(Directory Name)



→ Pancard.jpg → Gangs of Wass.

→ Selfies.jpg

getAllDirectory (Movies)
↳ return → Bestmovies
↳ worst

getAllFiles(Root)
↳ Pancard.jpg
↳ Selfies.jpg.

getAllDirector (Root)
↳ return → Person
↳ Learning
↳ Business

search (root, "file")

Selmon Boi Pic?

```
bool   Search( DirectoryName,   FileName)
{   //Assumption: Search(DN, FN) return true if FN is
                                 present anywhere inside DN
```

```
List<str> files = getAllFiles(DirectoryName);
 for (i=0; i< files.size ; i++)
 {
       if (files[i]== fileName)
            return True.

   }
```

Base Condition

```
List<str>  dir  =  getAllDirectories(DirectoryName)
 for(i=0 ; i<dir.size ; i++)
  {
        if ((search(dir[i], fileName)))
              return True ;

    }
  return false.
}
```

main logic

Doubts

Recursion