9/10/25 Class -> 1

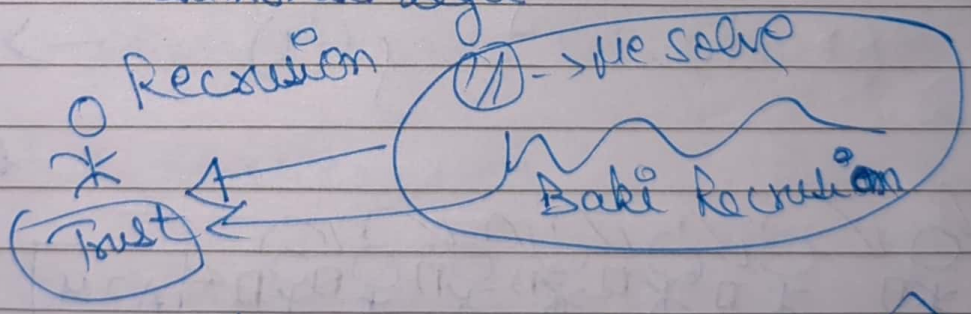
Date 103

Recursion -> When a function calls its self directly / indirectly

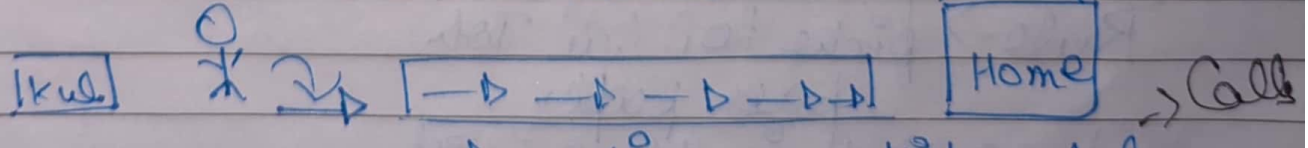
↳ In-dept -> Soln of Bigger Problem -> Soln of chotti problem of same type

```
Solve(){
    Solve()
}
```

Note :- 1 Case Tum solve kro baki recursion sambhal lega



Assum 1 step chalna ata hai



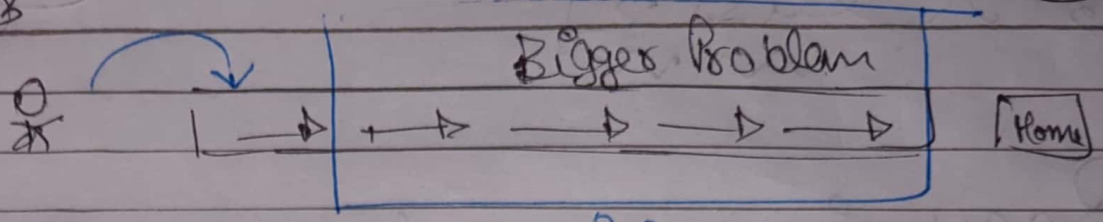
-> Calls

Recursion -> or kitna chalna.
Base Case -> Kab Rukna hai
Processing -> left / Right

Spiral

9/10/23

Date 104



Chotti Problem

↓↓

Recursion will solve it

Note → To understand Recursion, you need to first understand Recursion

f(6) → → → → → →

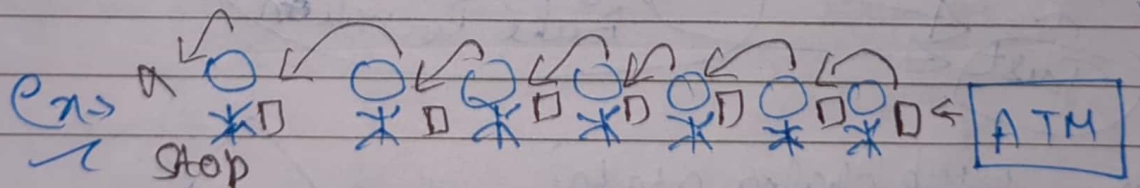
f(5) → → → → →

f(4) → → → →

f(3) → → →

f(2) → →

f(1) → (ek step aur chalna)



Rule → (i) Piche koi hai toh pass pakdo

(ii) agar koi nhi hai toh Rukh jao

Spiral

Date 105

9/10/23

1) → solve(n) → 2^n

Biggest → 2^n

solve(n) = 2^n

solve(n) = $2 \times 2^{n-1}$

| solve(n) = 2 × solve(n-1) |

Big ↑ chotti ↑

(ii) solve (n → 1) Counting

solve (n) ⇒ n, n-1, n-2, -- 1

solve (n) = n, (n-1), (n-2) -- -1

solve (n-1) = n-1, n-2, n-3 -- -1

⇒ solve(n) = n, solve(n-1)

(iii) 5! = 5 × 4 × 3 × 2 × 1

solve (5) = 5!

solve 5 = 5 × 4 × 3 × 2 × 1

solve (4) = 4 × 3 × 2 × 1

| solve(n) = n * solve(n-1) |

Bigger ↓ ye mujhe krna hai ↓ choti

Spiral

9/10/23

Recursion

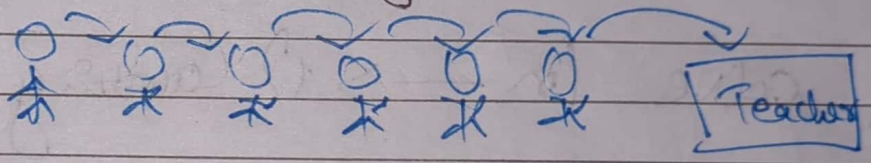
Date 106

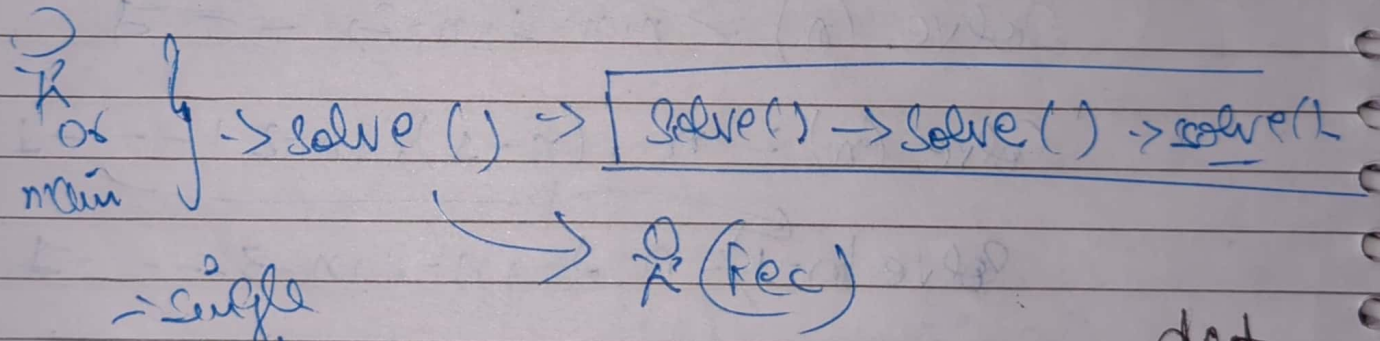
Rule -> if -> piche present pass paper

(ii) if -> piche koi nhi hai Rukh jao

ye humare dwara ki gayi function call

Rule -> aage wale ko ₹1 kar ke bhej do





Overall Ans = 1 + Recursion

9/10/23

Date 107

Factorial → $n! = n \times (n-1) \times (n-2) \cdots 3 \times 2 \times 1$

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

fact(7) = 7 × 6!

fact(7) = 7 × fact(6)

fact(n) = n * fact(n-1)

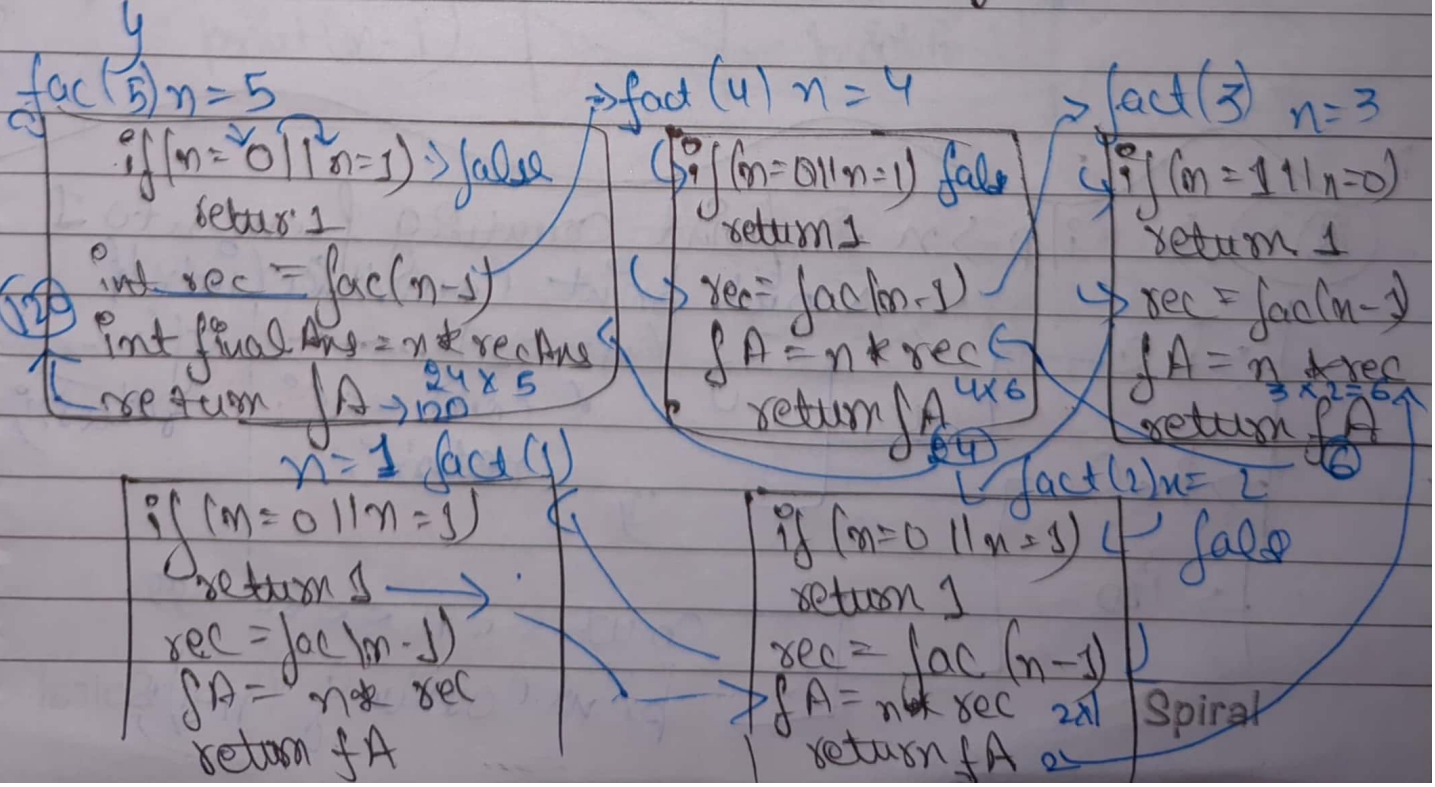
Base Case: if(n==1 || n==0) return 1;

Recursion → Base Case → Mandatory

solve(){ ↑↓ Recursive Call

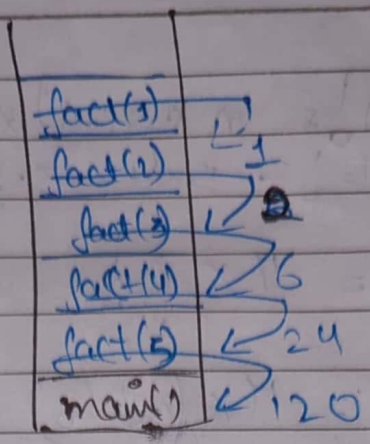
solve() } → Recursive Call

→ Processing → optional



Date 10.8

9/10/23

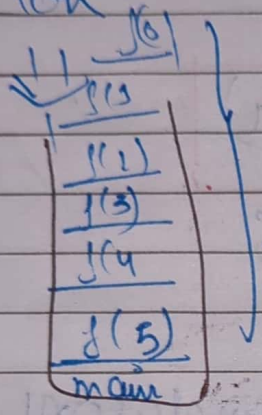


Base Case

↓↓

Nhi Hona

Toh



→ stack overflow pe crash & error

H/W → without Base Case Recursion Xe bitni Memory le ↳ Look in Task Manager

Q2 → i/p -> n → print counting from n to 1

```
void printCounting(int n){
    if (n == -1)
    { cout << 1;
      return;
    }
    if (n == 0)
    return;
    cout << n << 66 ;
    printCounting(n-1);
}
```

→ BC

→ R.C

→ Poo

Spiral

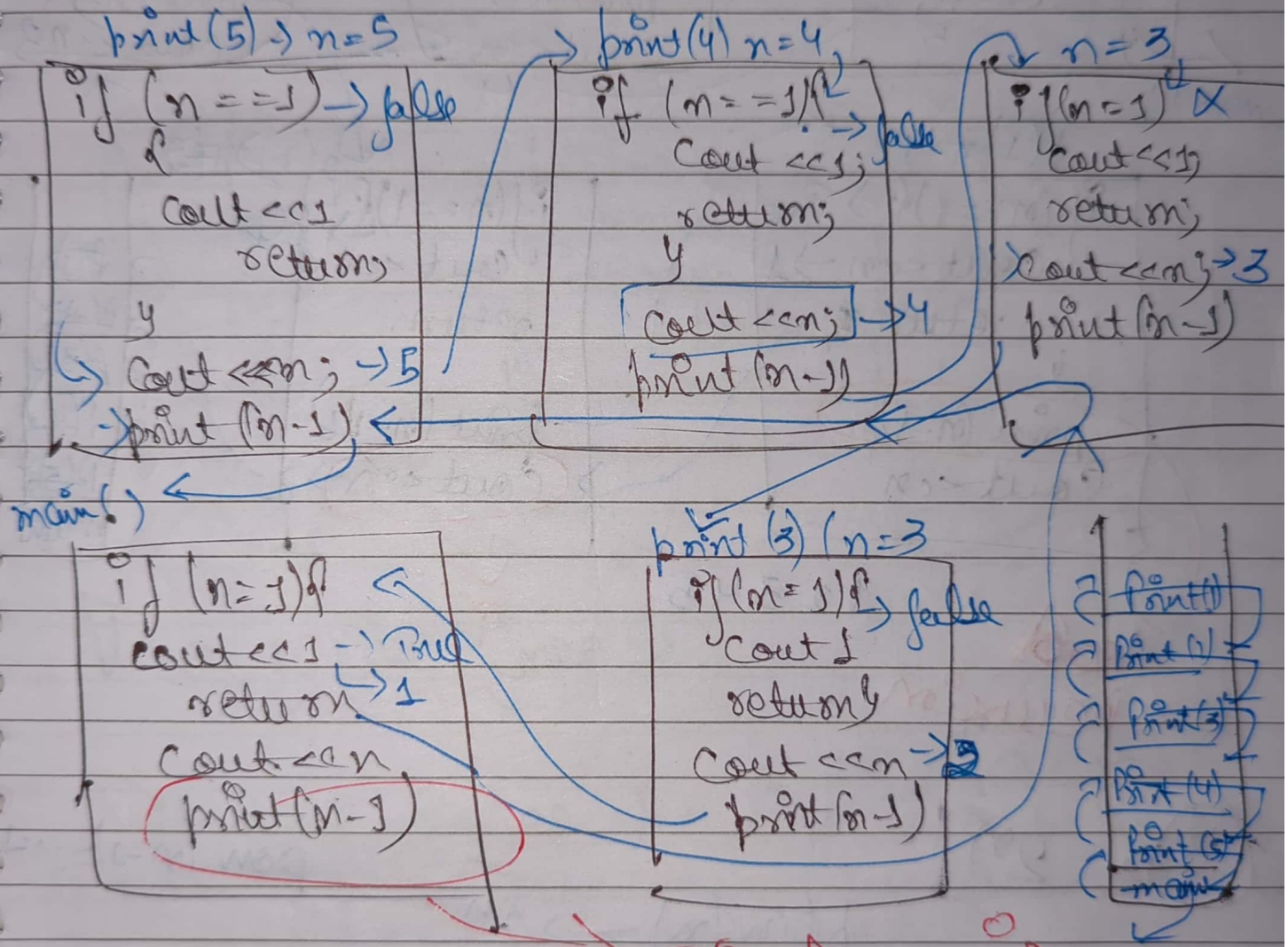
9/10/25

Date 109

Base case

Then → Recursive call ⇄ can be interchange

Then → processing



Tail Recursion

3 4 3 2 1

Head Recursion

Base Case

↓

Recursion Call

↓

Processing

Tail Recursion

Base Case

↓

processing

↓

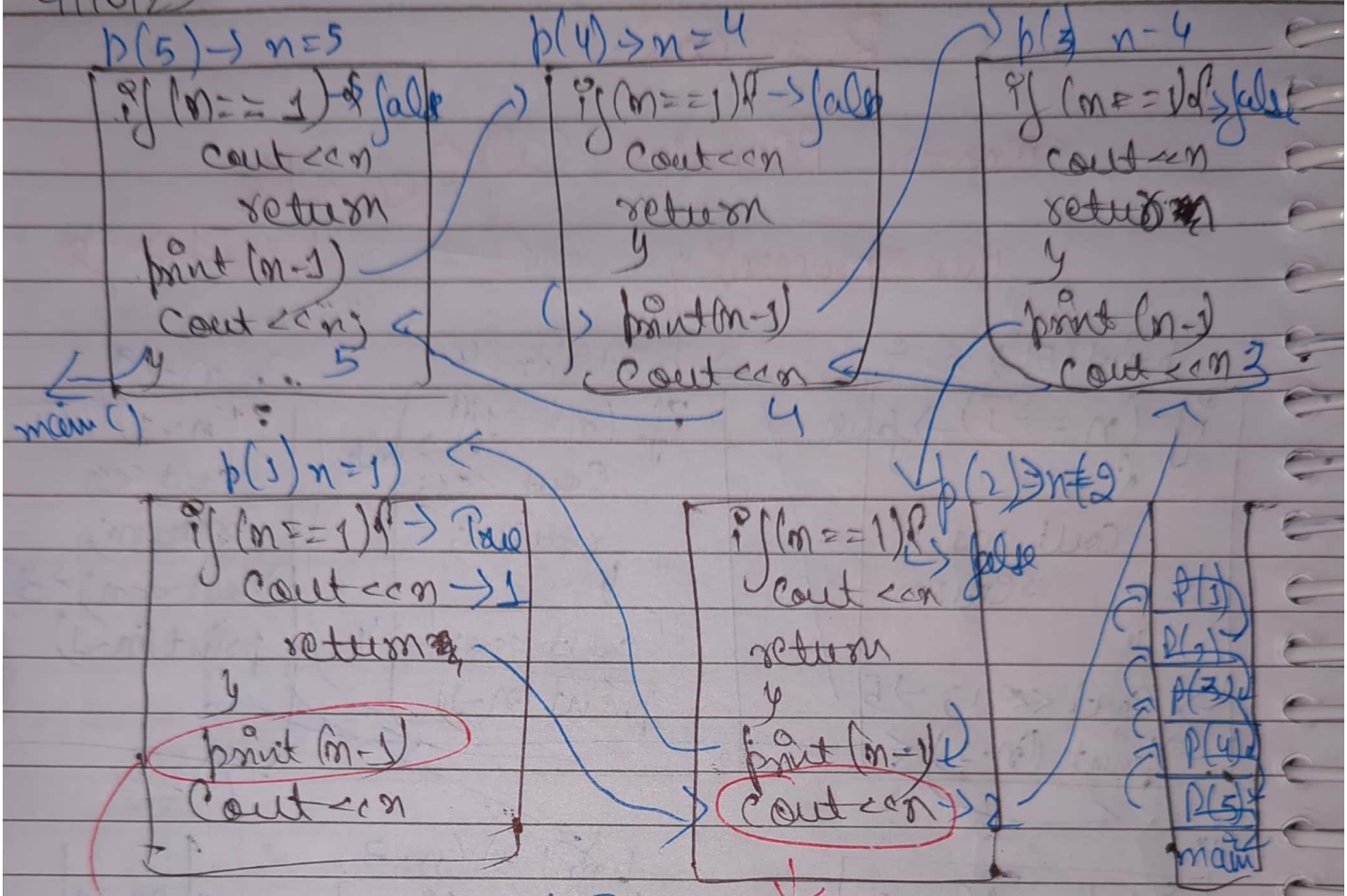
Recursion call

Spiral

Scanned with CamScanner

Date: 4-10

9/10/23



3) 2^n

pow(n) → 2^n

pow(n) = n x 2^{n-1}

pow(n) = 2 x pow(n-1)

pow(n-1) = 2^{n-1}

2^5 → 2^4 → 2^3 → 2^2 → 2^1 → 2^0

$2^{(n)}$

n = 0

return

Spiral

Date 111

9/10/23

pow(4) -> n(4

```
if n==0 -> false
    return 1;
int ans = 2* pow(n-1)
return ans;
```

8 x 2 = 16

pow(3) -> n=3

```
if(n==0) -> false
    return 1;
int ans = 2*
        pow(n-1);
return ans;
```

1 x 2 = 8

pow(2) = n=2

```
if(n==0) -> false
    return 1;
int ans = 2*
        pow(n-1);
return ans
```

4

pow(0) -> n=0

```
if(n==0) -> True
    return 1;
int ans = 2*
        pow(n-1);
return ans
```

pow(1) -> n=1

```
if(n==0) -> false
    return 1
int ans = 2*
        pow(n-1);
return ans
```

2

Pow(0)
Pow(1)
Pow(2)
Pow(3)
Pow(4)
main()

1, 2, 4, 8, 16

16

Recursive Tree →

Fibonacci Series → 0, 1, 1, 2, 3, 5, 8, 13, 21

```
if(n==0)
    return 0;
if(n==1)
    return ;
```

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

$$\boxed{f(n) = f(n-1) + f(n-2)}$$

Spiral

9/10/23

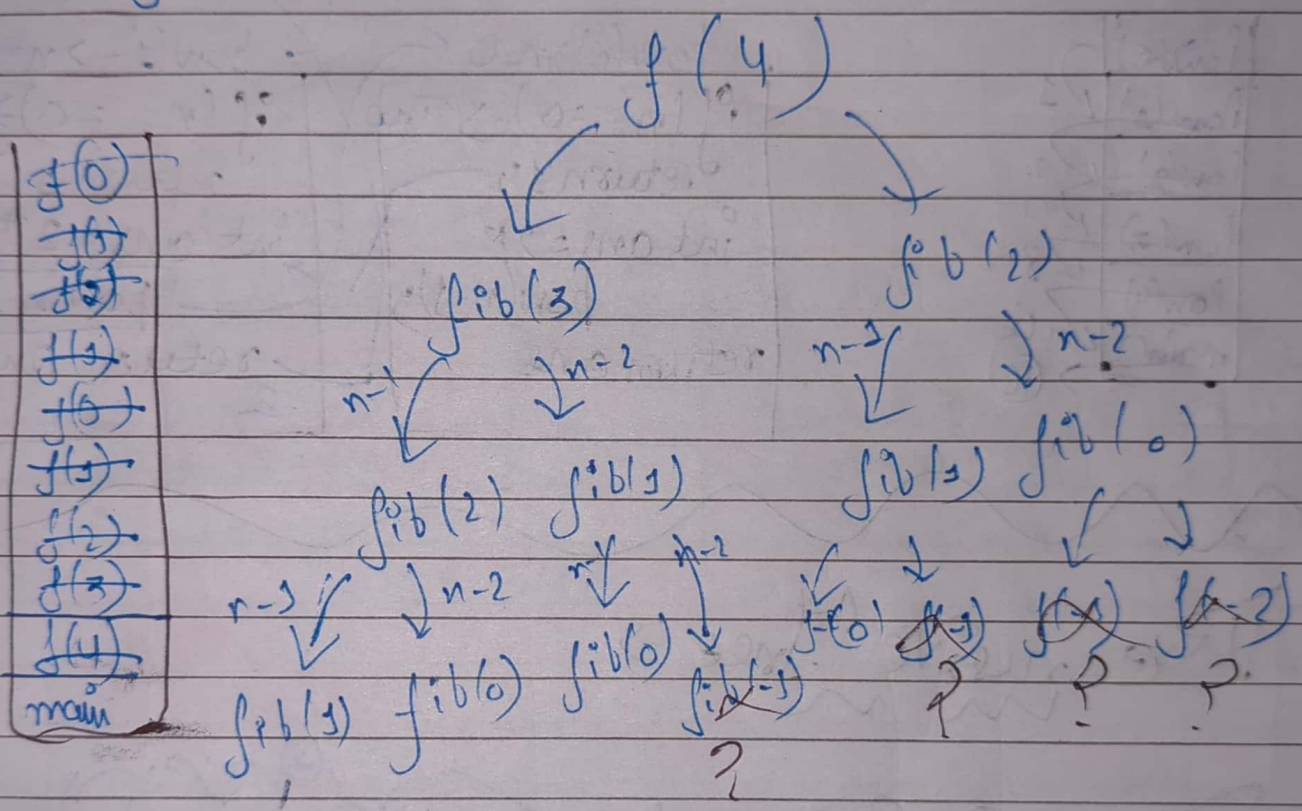
Date 112

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ——

0 1 2 3 4 5 6 7 8 9

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

$f(4) = f(3) + f(2)$ . . . .



O see Niche Nhi jaa pa Rha toh ans likh lo

1 -> Wasely ans likh lo

if (n == 0 || n == 1) → return 0 return 1

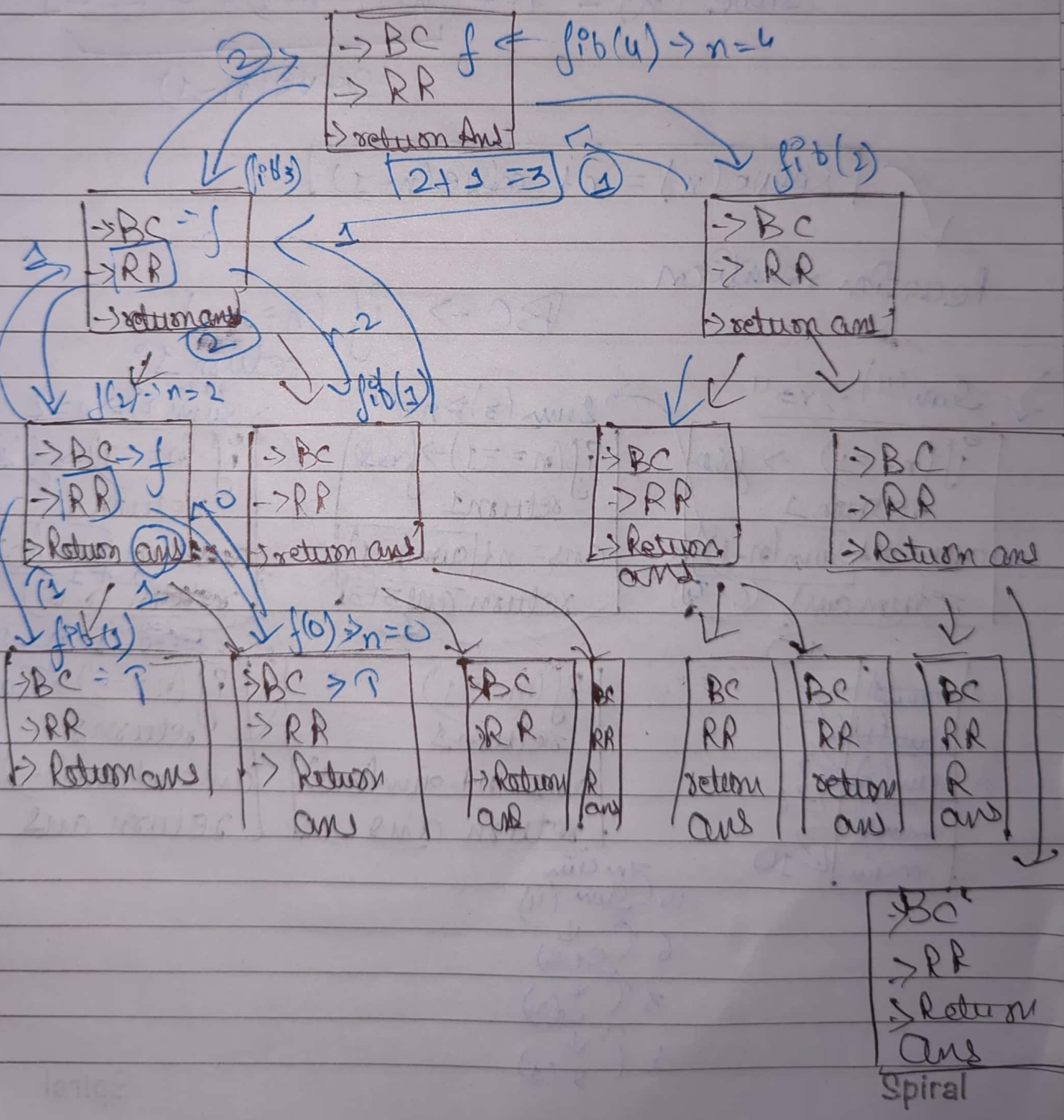
Spiral

9/10/23

Date 11 3

BC → if (n == 0) return 0; if (n == 1) return 1

RR → int ans = fib(n-1) + fib(n-2)



9/10/23

Date 164

Note : To understand Recursion, you need to first understand Recursion

Q) -> Sum -> n -> 1 print

Solve(n) = n + (n-1) + (n-2) + ---- 1

Solve(n-1)

Solve(n) = n + solve(n-1)

Recursion relation

BC -> if (n==1) return 1;

Sum(4) -> n=4

if(n==1) -> false
return 1
ans = n + Sum(n-1)
return ans (6+4)

10

Sum(3) => n=4

if(n==1) -> false
return 1
ans = n + sum(n-1)
return ans 3+3

6

Sum(2) => n=2

if(n==1) -> false
return 1
ans = n + sum(n-1)
2 + 1
return ans

3

n=1

if(n==1) -> True
return 1
ans = n + sum(n-1)
return ans

if(n==1)
return 1
ans = n + sum(n-1)
return ans

sum(4)
sum(3)
sum(2)
sum(1)
main

1
3
6
10

main
10 Sum(4)
6 S(3)
3 S(2)
1 S(1)

Spiral