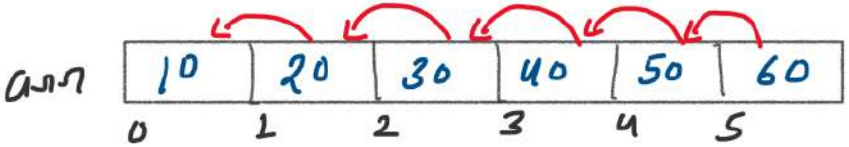
13 /10/2023

RECURSION CLASS3

✅Program 01: Check array sorted or not



Base case

```
if(index >= size)
    return True
```

Processing

```
if( arr[index] > arr[index-1])
 {
     f(arr, size, index+1);
 }
Else
  {
     return false;
  }
```

Relation

1st step

20 > 10 → Index = 1

30 > 20 → Index = 2

40 > 30 → Index = 3

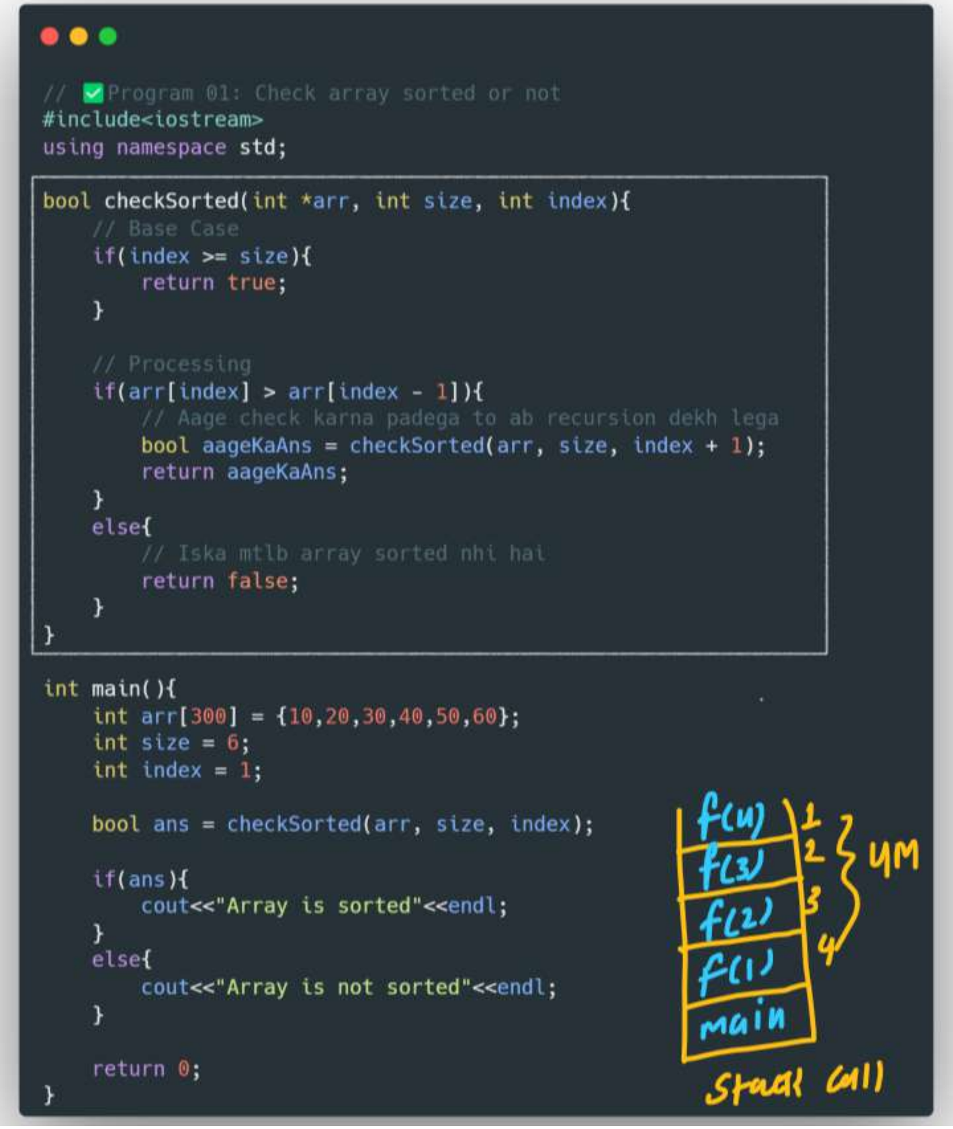
50 > 40 → Index = 4

60 > 50 → Index = 5

→ Index = 6 ✗ Ruk jao ...

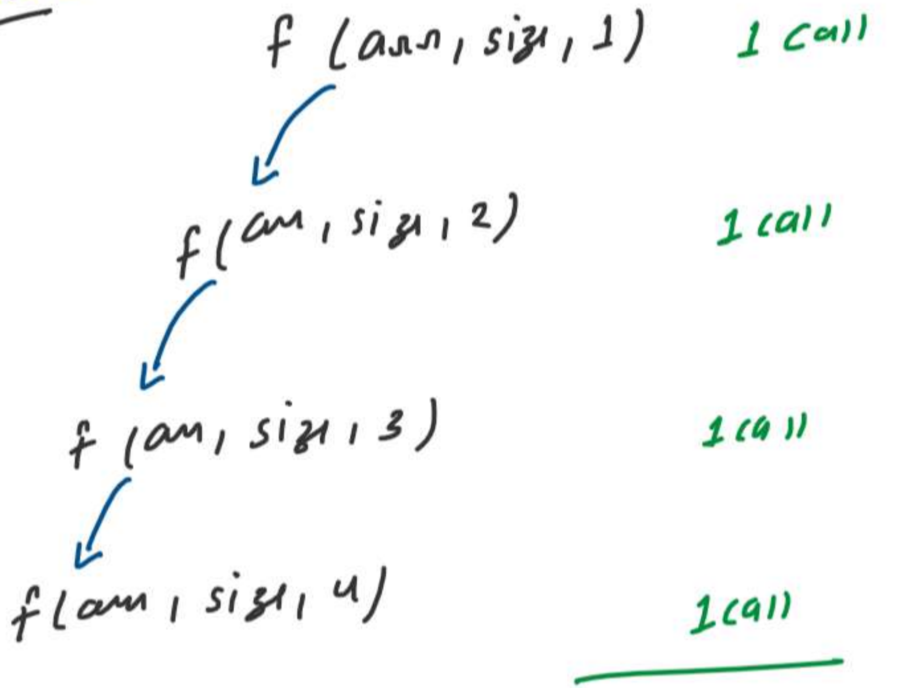
arr[index] >= arr[index-1]

Recursion Dekh lega



```cpp
// ✅Program 01: Check array sorted or not
#include<iostream>
using namespace std;

bool checkSorted(int *arr, int size, int index){
    // Base Case
    if(index >= size){
        return true;
    }

    // Processing
    if(arr[index] > arr[index - 1]){
        // Aage check karna padega to ab recursion dekh lega
        bool aageKaAns = checkSorted(arr, size, index + 1);
        return aageKaAns;
    }
    else{
        // Iska mtlb array sorted nhi hai
        return false;
    }
}

int main(){
    int arr[300] = {10,20,30,40,50,60};
    int size = 6;
    int index = 1;

    bool ans = checkSorted(arr, size, index);

    if(ans){
        cout<<"Array is sorted"<<endl;
    }
    else{
        cout<<"Array is not sorted"<<endl;
    }

    return 0;
}
```

Size = N = 4



Total call = 4

T.C = O(N)

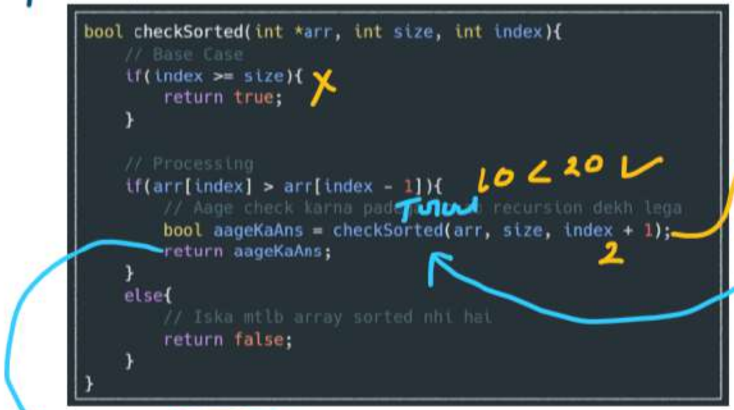
T(N) = O(N)

S.C = O(N)

→ O(4M) ⇒ O(NM) → Byte (constant)

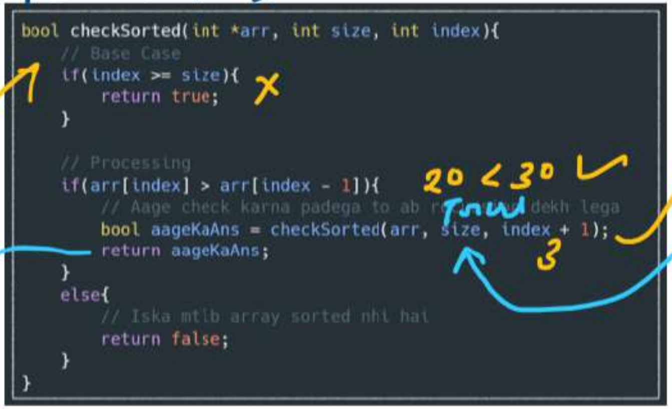
⇒ O(N)

f(arr, 3, 1)



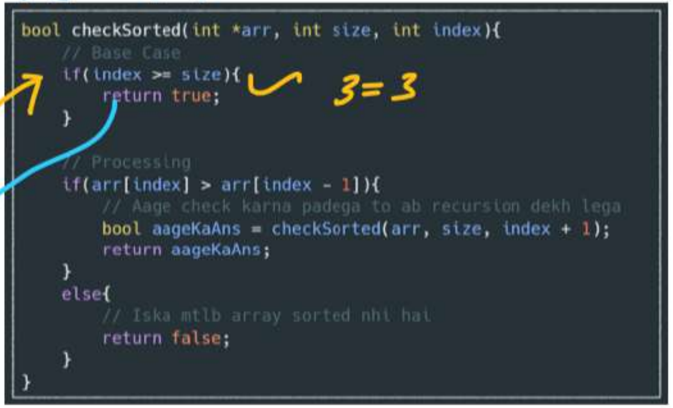
```
bool checkSorted(int *arr, int size, int index){
    // Base Case
    if(index >= size){
        return true;
    }

    // Processing
    if(arr[index] > arr[index - 1]){
        // Aage check karna padega to ab recursion dekh lega
        bool aageKaAns = checkSorted(arr, size, index + 1);
        return aageKaAns;
    }
    else{
        // Iska mtlb array sorted nhi hai
        return false;
    }
}
```

f(arr, 3, 2)



```
bool checkSorted(int *arr, int size, int index){
    // Base Case
    if(index >= size){
        return true;
    }

    // Processing
    if(arr[index] > arr[index - 1]){
        // Aage check karna padega to ab recursion dekh lega
        bool aageKaAns = checkSorted(arr, size, index + 1);
        return aageKaAns;
    }
    else{
        // Iska mtlb array sorted nhi hai
        return false;
    }
}
```

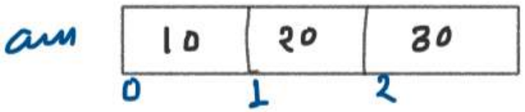
f(arr, 3, 3)



```
bool checkSorted(int *arr, int size, int index){
    // Base Case
    if(index >= size){
        return true;
    }

    // Processing
    if(arr[index] > arr[index - 1]){
        // Aage check karna padega to ab recursion dekh lega
        bool aageKaAns = checkSorted(arr, size, index + 1);
        return aageKaAns;
    }
    else{
        // Iska mtlb array sorted nhi hai
        return false;
    }
}
```

TRUE

SORTED ARRAY



f(arr, size, index)

MAIN

✅Program 02: Binary search recursive solution

Target = 40

arr

			MID			
10	20	30	40	50	60	70
0	1	2	3	4	5	6

Base case

```
if (s > e)
    return -1
```

Processing

```
if (arr[mid] > Target)
    Return f(arr, Target, start, mid-1);
Else
    return f(arr, target, mid+1, end);
```

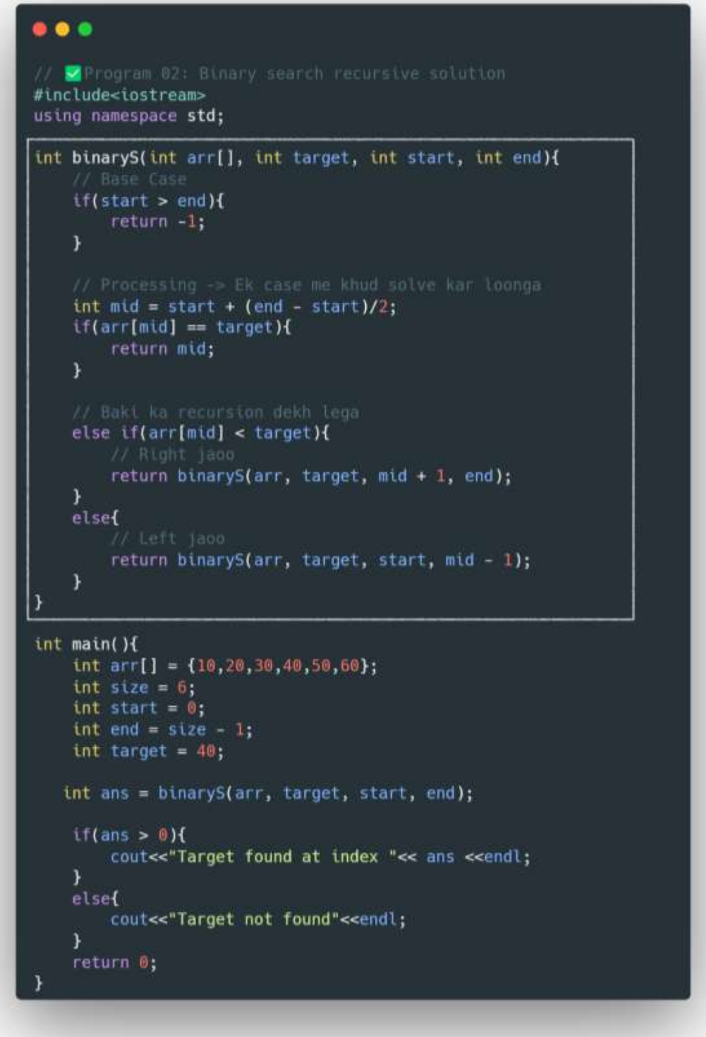
Relation

Start = 0   End = 6 $mid = \frac{start + End}{2}$

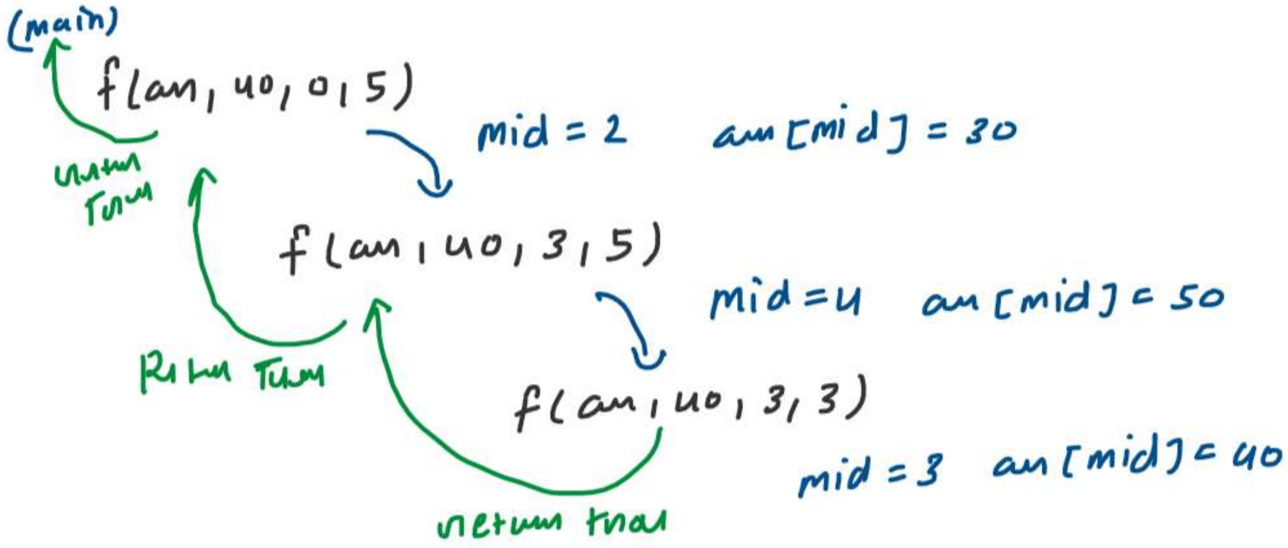
1st STEP → if (arr[mid] == Target)
↳ Return mid

Recursion dekh lega

```
if (arr[mid] > Target)
    ↳ return f(arr, Target, start, mid-1);
Else
    ↳ return f(arr, Target, mid+1, End);
```



```cpp
// ✅Program 02: Binary search recursive solution
#include<iostream>
using namespace std;

int binaryS(int arr[], int target, int start, int end){
    // Base Case
    if(start > end){
        return -1;
    }

    // Processing -> Ek case me khud solve kar loonga
    int mid = start + (end - start)/2;
    if(arr[mid] == target){
        return mid;
    }

    // Baki ka recursion dekh lega
    else if(arr[mid] < target){
        // Right jaoo
        return binaryS(arr, target, mid + 1, end);
    }
    else{
        // Left jaoo
        return binaryS(arr, target, start, mid - 1);
    }
}

int main(){
    int arr[] = {10,20,30,40,50,60};
    int size = 6;
    int start = 0;
    int end = size - 1;
    int target = 40;

   int ans = binaryS(arr, target, start, end);

    if(ans > 0){
        cout<<"Target found at index "<< ans <<endl;
    }
    else{
        cout<<"Target not found"<<endl;
    }
    return 0;
}
```



T.C. and S.C. ⇒ O(logN)

✅Program 03: Subsequence of string

Pattern 01: In$\overset{✓}{c}$lude and ex$\overset{✗}{c}$lude pattern

Ex-1

String = "ABC";

A	B	C	
✓	✗	✗	= "A"
✗	✓	✗	= "B"
✗	✗	✓	= "C"
✓	✓	✗	= "AB"
✓	✗	✓	= "AC"
✗	✓	✓	= "BC"
✓	✓	✓	= "ABC"
✗	✗	✗	= " "

Total sub-sequence string are 8 $\boxed{2^3}$

String size = N = 3

Ex-2

String = "xy"

✓	✗	= "x"
✗	✓	= "y"
✓	✓	= "xy"
✗	✗	= " "

Total SS = 4

= $\boxed{2^2}$

$\boxed{N = 2}$

str = "ABC"

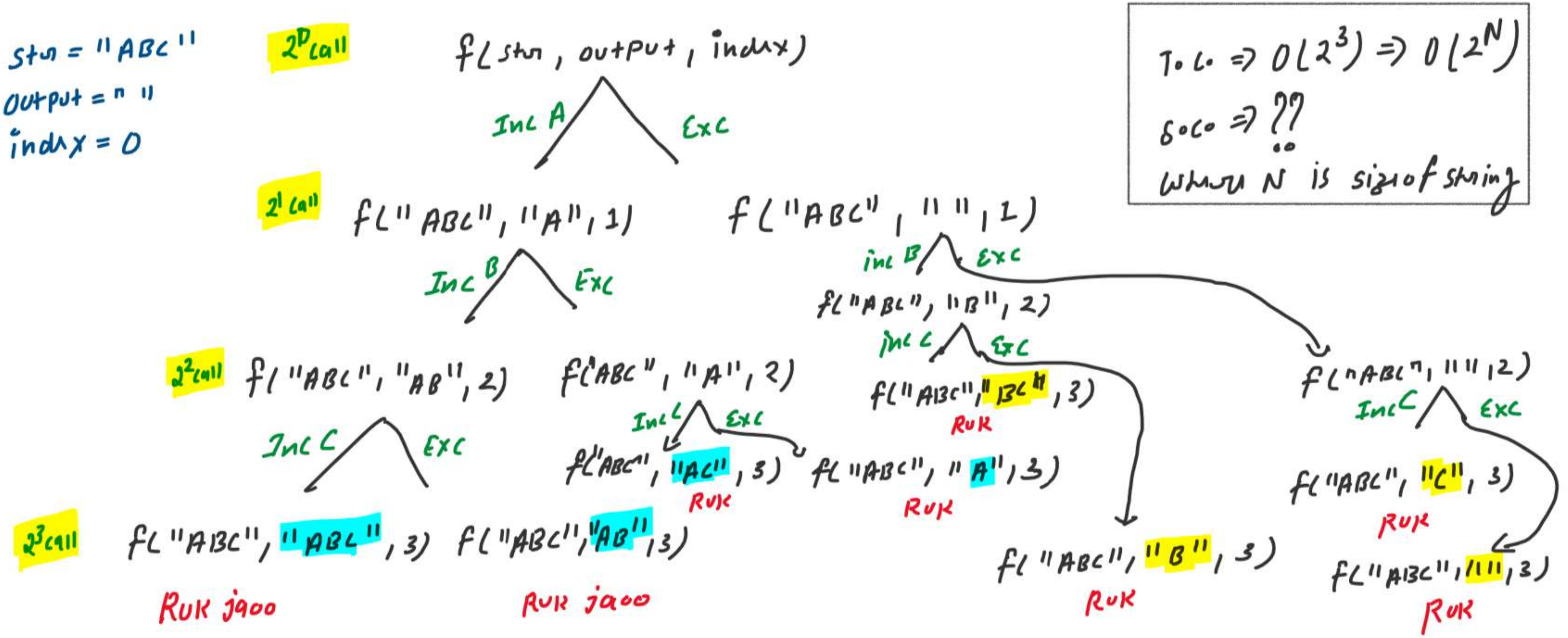
output = " "

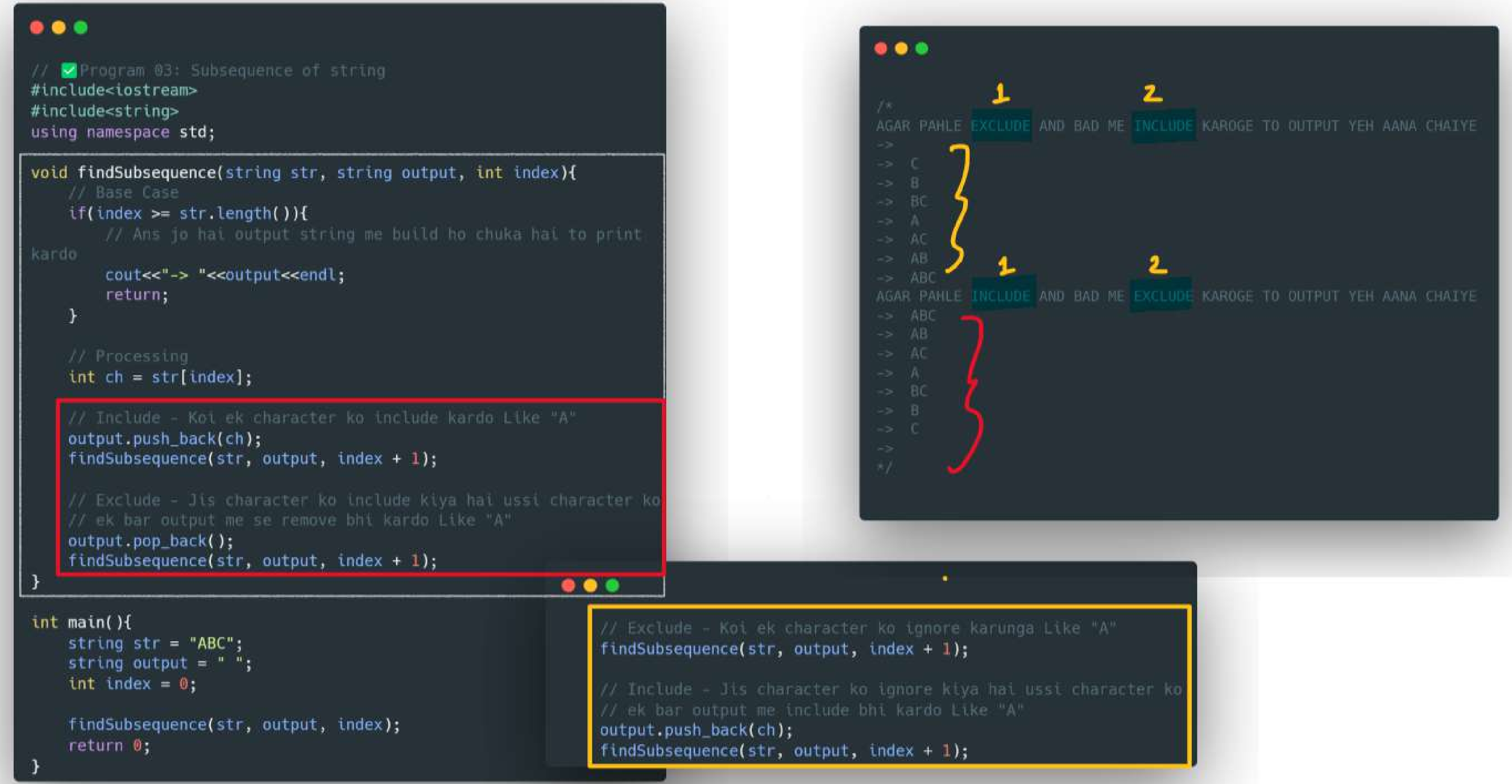
index = 0

T.C. => $O(2^3)$ => $O(2^N)$

S.C. => ??

where N is sizeof string





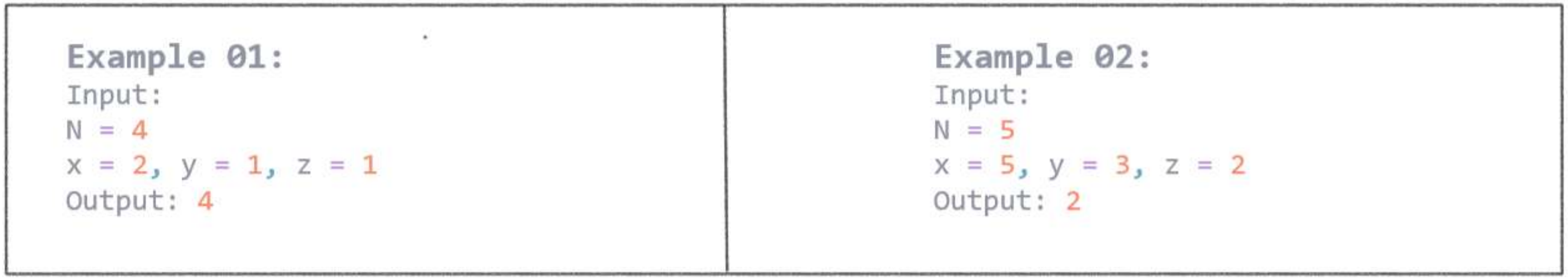
```cpp
// ✅Program 03: Subsequence of string
#include<iostream>
#include<string>
using namespace std;

void findSubsequence(string str, string output, int index){
    // Base Case
    if(index >= str.length()){
        // Ans jo hai output string me build ho chuka hai to print
kardo
        cout<<"-> "<<output<<endl;
        return;
    }

    // Processing
    int ch = str[index];

    // Include - Koi ek character ko include kardo Like "A"
    output.push_back(ch);
    findSubsequence(str, output, index + 1);

    // Exclude - Jis character ko include kiya hai ussi character ko
    // ek bar output me se remove bhi kardo Like "A"
    output.pop_back();
    findSubsequence(str, output, index + 1);
}

int main(){
    string str = "ABC";
    string output = " ";
    int index = 0;

    findSubsequence(str, output, index);
    return 0;
}
```

```
/*
AGAR PAHLE EXCLUDE AND BAD ME INCLUDE KAROGE TO OUTPUT YEH AANA CHAIYE
->
->  C
->  B
->  BC
->  A
->  AC
->  AB
->  ABC
AGAR PAHLE INCLUDE AND BAD ME EXCLUDE KAROGE TO OUTPUT YEH AANA CHAIYE
->  ABC
->  AB
->  AC
->  A
->  BC
->  B
->  C
->
*/
```

```cpp
// Exclude - Koi ek character ko ignore karunga Like "A"
findSubsequence(str, output, index + 1);

// Include - Jis character ko ignore kiya hai ussi character ko
// ek bar output me include bhi kardo Like "A"
output.push_back(ch);
findSubsequence(str, output, index + 1);
```

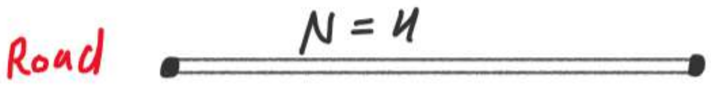
✓Program 04: Maximize the cost segment (GFG)

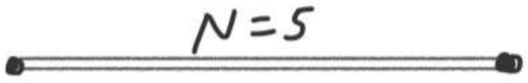
Pattern 02: Exploring all possible ways pattern

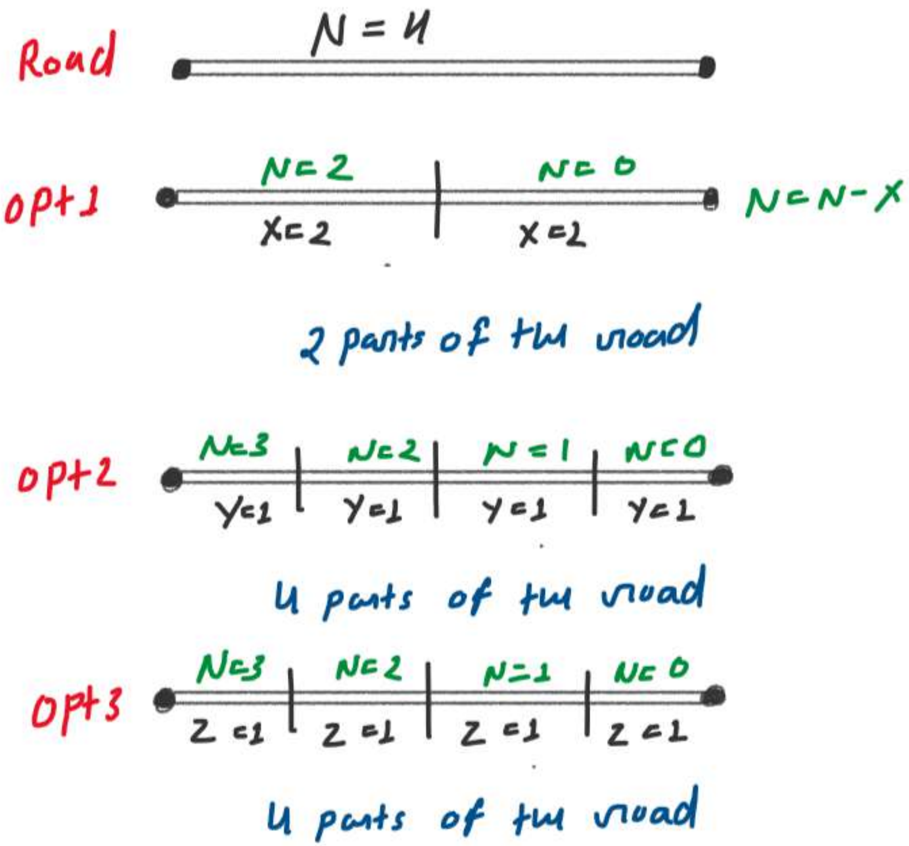


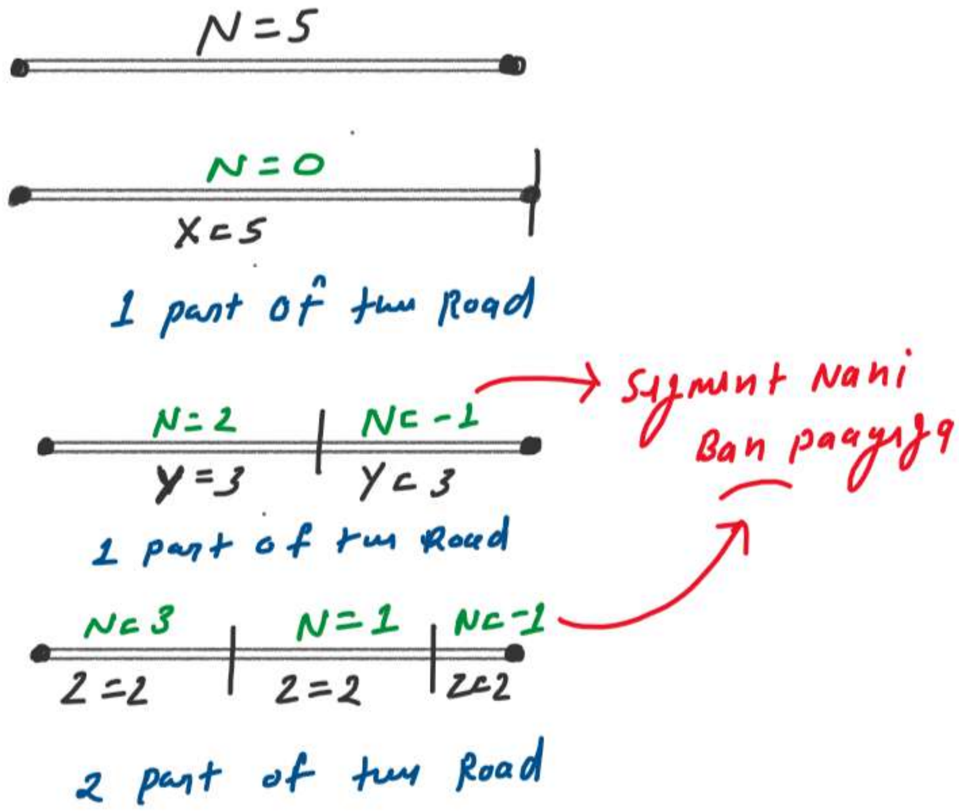
Example 01:
Input:
N = 4
x = 2, y = 1, z = 1
Output: 4

Example 02:
Input:
N = 5
x = 5, y = 3, z = 2
Output: 2



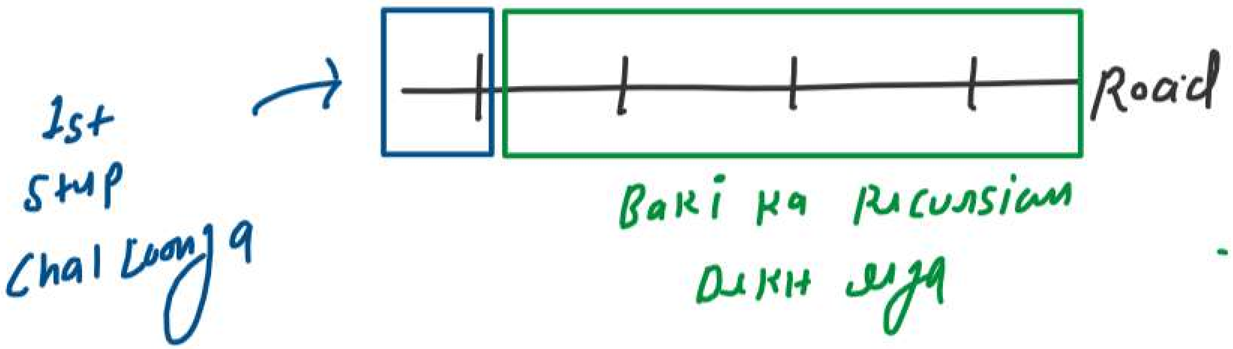






Max(opt1, opt2, op3) = max(2, 4, 4)
= 4 output

max(1, 1, 2) = 2 output



Relation

option1 = $1 + f(N-x, x, y, z)$

option2 = $1 + f(N-y, x, y, z)$

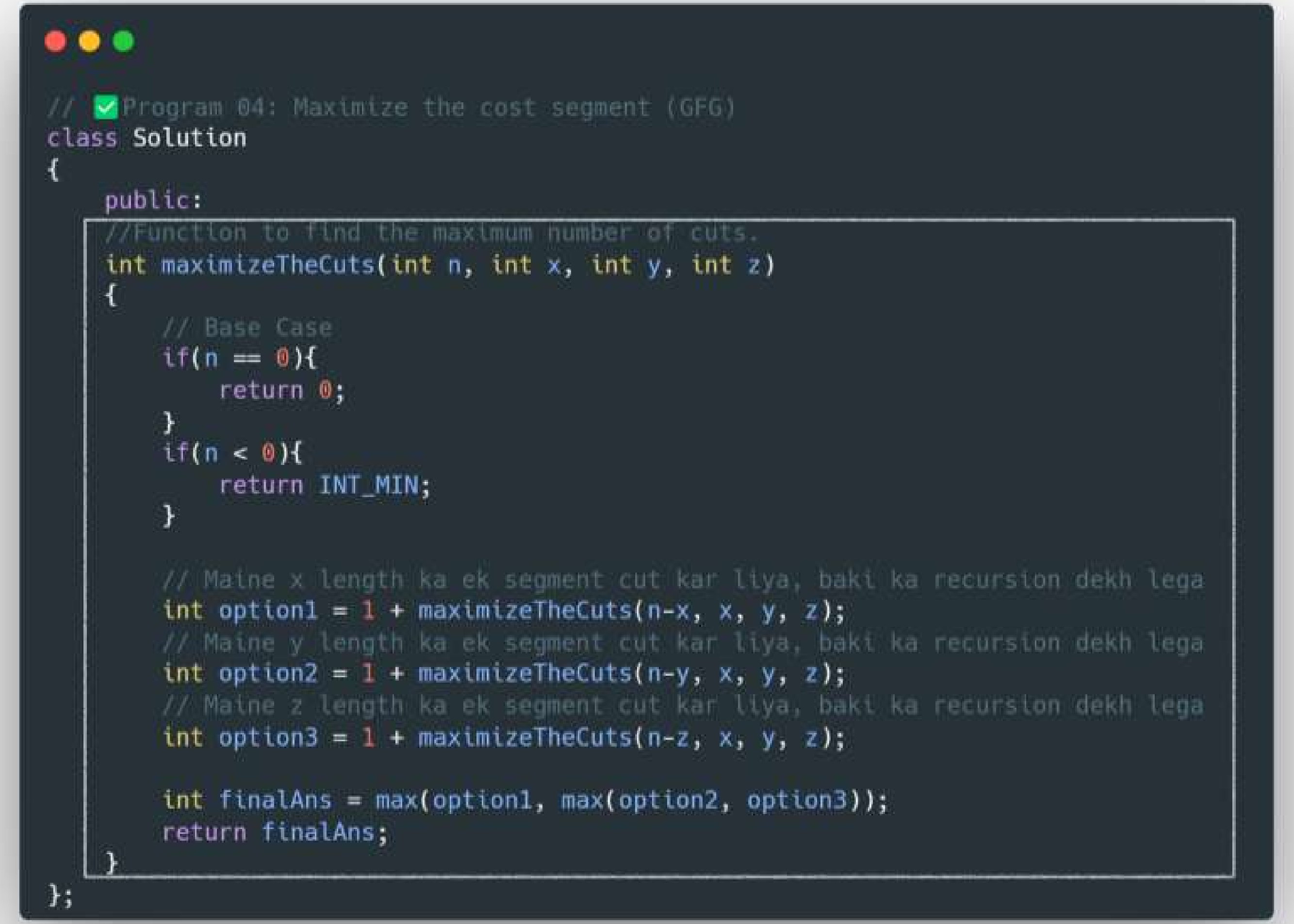
option3 = $1 + f(N-z, x, y, z)$

BASE case

```
if(N==0)
    return 0;

if(N<0)
    return invalid no.
```

→ Agar N=0 hai to zero segment ban gaye honge

→ Agar N<0 to iss case main koi bhi segment nahi banna chaiye



```cpp
// ✅Program 04: Maximize the cost segment (GFG)
class Solution
{
    public:
    //Function to find the maximum number of cuts.
    int maximizeTheCuts(int n, int x, int y, int z)
    {
        // Base Case
        if(n == 0){
            return 0;
        }
        if(n < 0){
            return INT_MIN;
        }

        // Maine x length ka ek segment cut kar liya, baki ka recursion dekh lega
        int option1 = 1 + maximizeTheCuts(n-x, x, y, z);
        // Maine y length ka ek segment cut kar liya, baki ka recursion dekh lega
        int option2 = 1 + maximizeTheCuts(n-y, x, y, z);
        // Maine z length ka ek segment cut kar liya, baki ka recursion dekh lega
        int option3 = 1 + maximizeTheCuts(n-z, x, y, z);

        int finalAns = max(option1, max(option2, option3));
        return finalAns;
    }
};
```

I will Do latur

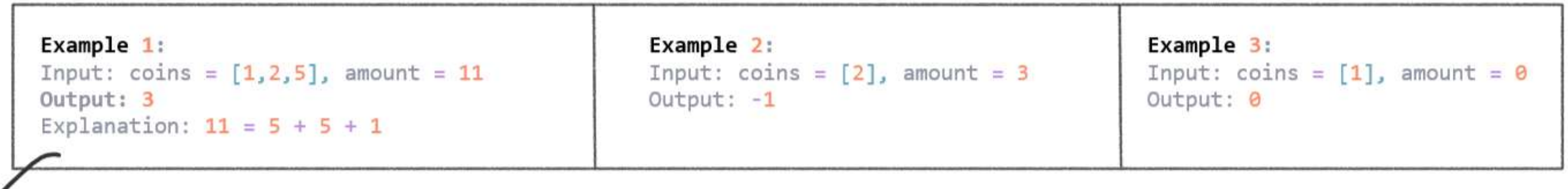
DRY RUN = ??

T.C = ??

S.C = ??

✅Program 05: Coin change (Leetcode-322)

Pattern 02: Exploring all possible ways pattern



Example 1:	Example 2:	Example 3:
Input: coins = [1,2,5], amount = 11 Output: 3 Explanation: 11 = 5 + 5 + 1	Input: coins = [2], amount = 3 Output: -1	Input: coins = [1], amount = 0 Output: 0

way1

Coin =1 Takes 11 times => 11 × 1 = 11   11 coins

Coin = 2 Takes 5 times & Coin = 1 takes 1 time => (5 × 2) + 1 = 11   6 Coins

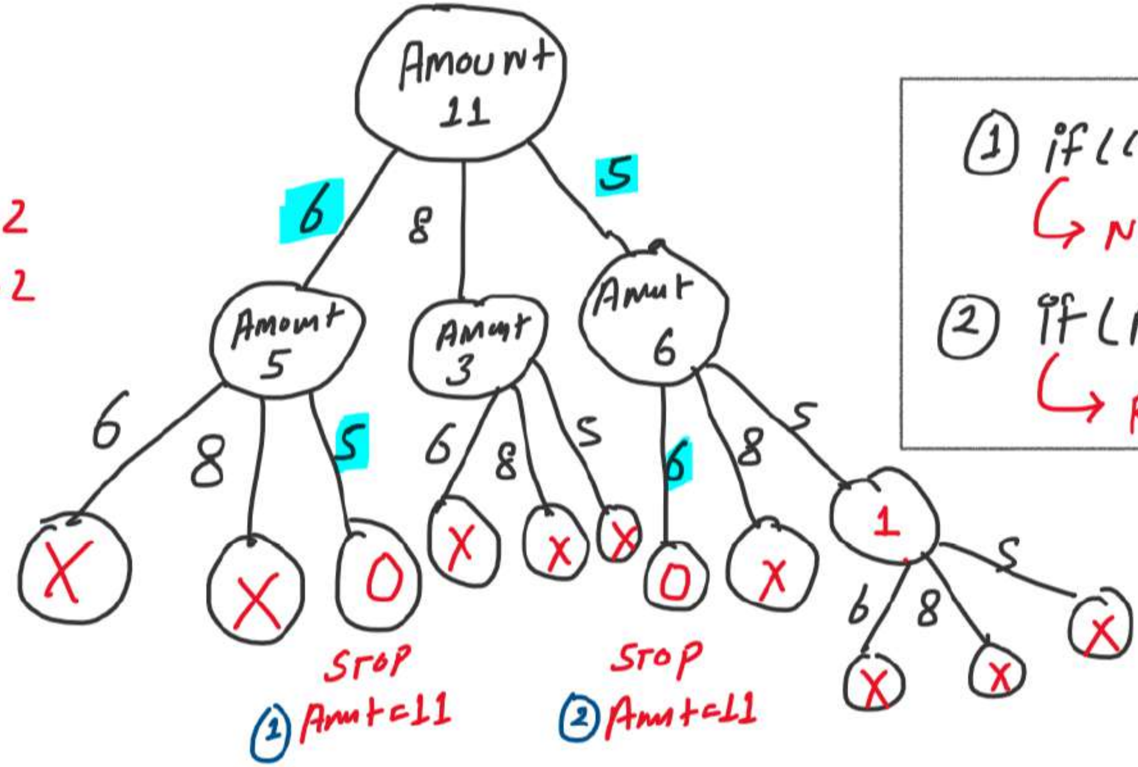
Coin = 5 Takes 2 times & Coin = 1 takes 1 time => (2 × 5) + 1 = 11   3 Coins

Min (11 , 6 , 3) => 3 coins

Way 2
Amount = 11
Coins = 6, 8, 5

Call
① 6 + 5 = 11 } coins = 2
② 5 + 6 = 11 } coins = 2

min(2, 2) = 2
Output



① if (coin > Amount)
↳ No Recursion call

② if (Amount == 0) ↳ Base case
↳ Return 0 coin

way:3 Amount = N = 11

Coins [6, 8, 5]

$f(N) = f(11) \xrightarrow{6\ coin} f(5) \Rightarrow \boxed{\boxed{1} + f(N-6)}$

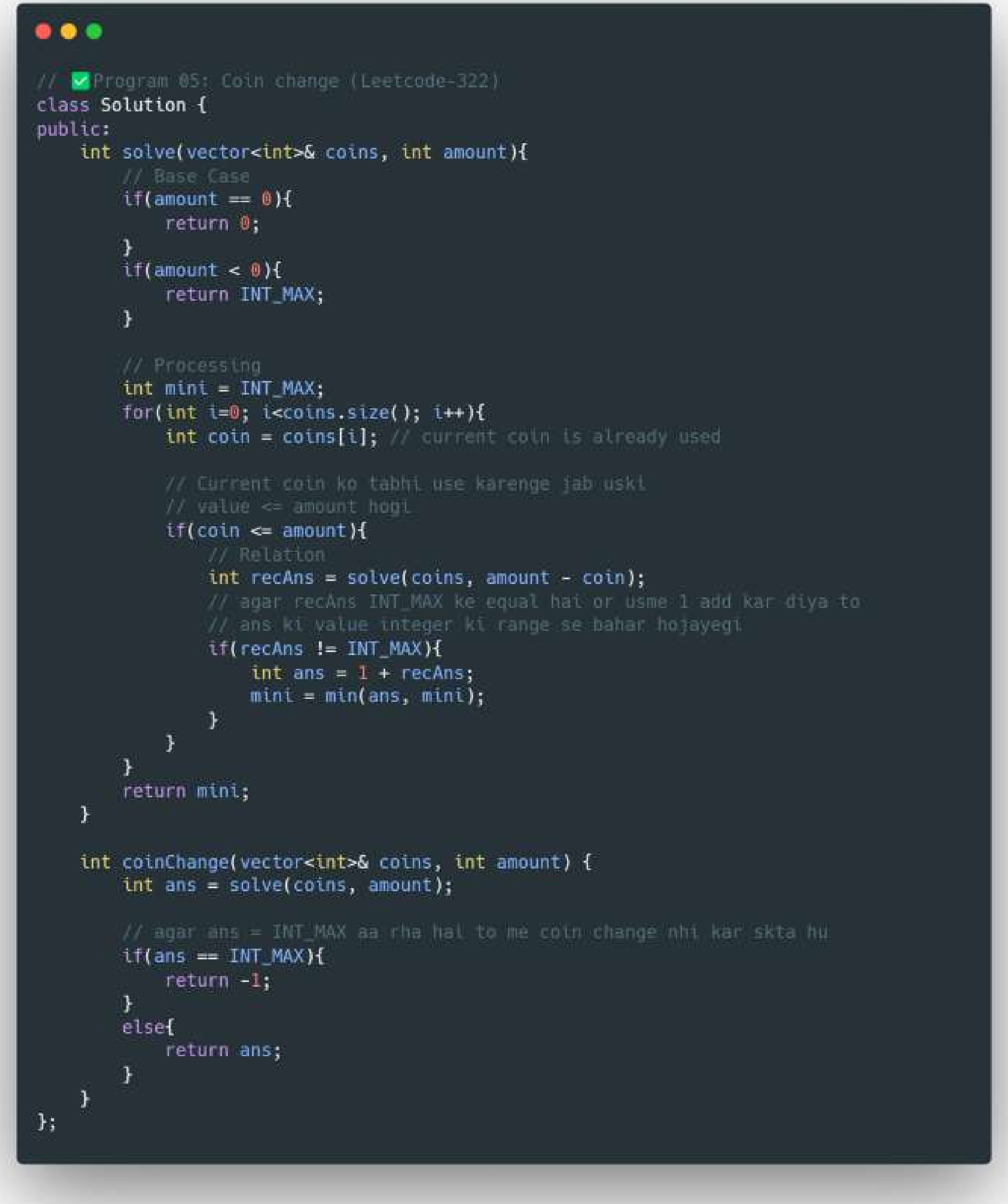
1 STEP

$f(N) = f(11) \xrightarrow{8\ coin} f(3) \Rightarrow \boxed{1 + f(N-8)}$

min

$f(N) = f(11) \xrightarrow{5\ coin} f(6) \Rightarrow \boxed{1 + f(N-5)}$

{ But AISA zarooni Nahi Hai Ki Har Bar 3 Types ka Coin Hi Ho USSE jada Bhi Ho Skata Hai To Har Ek Coin Ko Check Karne ke Leye Loop ka USe Karenge. KI (coin<= Amount) Hai ya Nahi }



```cpp
// ✅Program 05: Coin change (Leetcode-322)
class Solution {
public:
    int solve(vector<int>& coins, int amount){
        // Base Case
        if(amount == 0){
            return 0;
        }
        if(amount < 0){
            return INT_MAX;
        }

        // Processing
        int mini = INT_MAX;
        for(int i=0; i<coins.size(); i++){
            int coin = coins[i]; // current coin is already used

            // Current coin ko tabhi use karenge jab uski
            // value <= amount hogi
            if(coin <= amount){
                // Relation
                int recAns = solve(coins, amount - coin);
                // agar recAns INT_MAX ke equal hai or usme 1 add kar diya to
                // ans ki value integer ki range se bahar hojayegi
                if(recAns != INT_MAX){
                    int ans = 1 + recAns;
                    mini = min(ans, mini);
                }
            }
        }
        return mini;
    }

    int coinChange(vector<int>& coins, int amount) {
        int ans = solve(coins, amount);

        // agar ans = INT_MAX aa rha hai to me coin change nhi kar skta hu
        if(ans == INT_MAX){
            return -1;
        }
        else{
            return ans;
        }
    }
};
```

Dry RUN

T.C. = ?

S.C. = ?

✅Program 06: House Robber (Leetcode-198)

Ex:01

Nums	4	5	7	1	2
	0	1	2	3	4

Amount of money

Nums[4] = 2

House No = 5

i = 4

Find Kya Karna Hai?

Return the maximum Amount of money

Case:1
Chori Karne wala Ghar

✓0th ~~1th~~
✓2th ~~3th~~ ~~1th~~
✓4th ~~3th~~

Amount
4
7
2
Total = 13

Case:2
Chori na Karne wala Ghar

✓1th ✗0th ✗2th
✓3th ✗2th ✗4th

Amount
5
1
Total = 6

$\boxed{\text{Max Amou = 13}}$ output

Example:02

NUMS	1	2	3	1
	0	1	2	3

Case-1

Chori Kane wala Ghar			Amount
✓0th	✗1th		1
✓2th	~~1~~th	✗3th	3
			4

Case-2

Choni Na Karne wala Ghar			Amount
✓1th	✗0th	✗2th	2
✓3th	✗2th		1
			3

$\boxed{\text{maxAmaut} = 4}$ output

NUMS	4	5	7	1	2
	0	1	2	3	4

Case:1
0th ghar me chori karunga

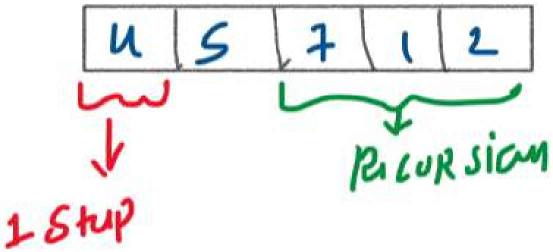
OR

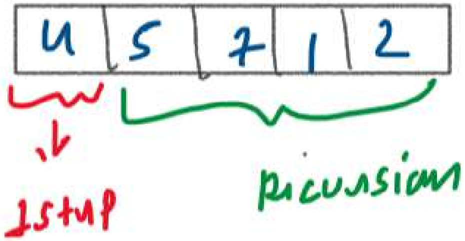
Case:2
0th ghar me chori nahi karunga

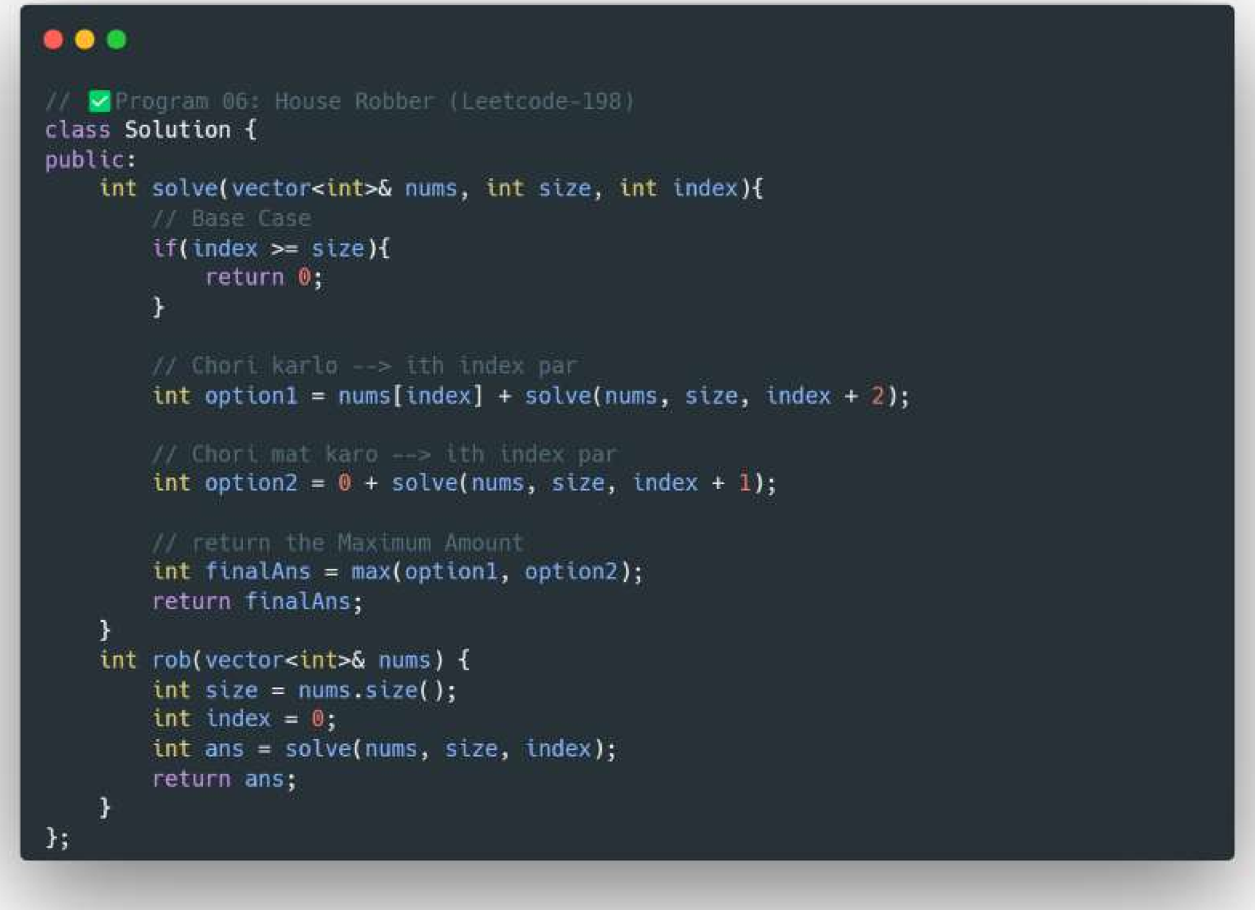
$4 + f(i+2, N-1)$

$0 + f(i+1, N-1)$

$\Big\}$ maxAmount ⇒ ?







```
// ✅Program 06: House Robber (Leetcode-198)
class Solution {
public:
    int solve(vector<int>& nums, int size, int index){
        // Base Case
        if(index >= size){
            return 0;
        }

        // Chori karlo --> ith index par
        int option1 = nums[index] + solve(nums, size, index + 2);

        // Chori mat karo --> ith index par
        int option2 = 0 + solve(nums, size, index + 1);

        // return the Maximum Amount
        int finalAns = max(option1, option2);
        return finalAns;
    }
    int rob(vector<int>& nums) {
        int size = nums.size();
        int index = 0;
        int ans = solve(nums, size, index);
        return ans;
    }
};
```

DRY RUN

T.C. = ?

S.C. = ?