

<b>Completed</b>	<b>At</b>
<b>1) Into to programming -</b>	<b>02</b>
<b>2) Arrays -</b>	<b>73</b>
<b>3) Sorting Algorithms -</b>	<b>132</b>
<b>4) Matrix in CPP –</b>	<b>158</b>
<b>5) Character Array &amp; Strings in CPP –</b>	<b>196</b>
<b>6) STL – Vector</b>	<b>222</b>
<b>7) Bit Manipulations –</b>	<b>249</b>
<b>8) OOP's in CPP –</b>	<b>277</b>
<b>9) Recursion –</b>	<b>350</b>
<b>10) Divide &amp; Conquer –</b>	<b>390</b>

## **[1] - Into to programming -**

1) Basic Into to programming and Fundamentals - Conditional & Looping Statement, Break & Continue, Functions, Pointers

2) digit sum -

3) Reverse Number -

4) Prime no. -

5) Nested Loops & Patterns -

5.2) Star Pattern

5.3) Inverted Star Pattern

5.4) half Pyramid Pattern

5.5) Character Pyramid Pattern

5.6) Hollow Rectangle Pattern

5.7) Inverted & Rotated Half Pyramid

5.8) Floyd's Triangle

5.9) Diamond Pattern

5.10) Butterfly Pattern

6) Number System -

6.1) Binary to Decimal Conversion -

6.2) Decimal to Binary -

## **[2] - Arrays-**

- 1) Initiation Methods of an array and Output & input of an array -
- 2) Largest Value/Max Element in the array & 2.1) Min Element -
- 3) Dereferences of Pointer concept in Array -
- 4) Linear Search in array -
- 5) Reverse an array -
  - 5.1) using extra space
  - 5.2 - W/o using Extra Space
- 6) Binary Search - It always for the Sorted Array
- 7) Array Pointer -
  - 7.1. Pointer Arithmetic Approaches -
  - 7.2 - Addition & Subtraction of Constants -
  - 7.3- Addition & Subtraction of Pointers
  - 7.4 - Comparison Operators
- 8) Subarrays -
- 9) Max Subarray Sum -
  - 9.1 - Clearly Brute Force Approach -
  - 9.2 - Slightly Optimized approach - for decreasing the time complexity
  - 9.3 - Using Most Optimized Approach - Kadane's Algorithms. For very less Time Complexity
- 10) buy & sell stock problem-
- 11) Trapping Rainwater Problem -

### **[3] - Sorting Algorithms -**

- 1) Bubble Sort Algo -
- 2) Selection Sort Implementation -
- 3) Insertion Sort Algorithm & Inbuilt Sorting technique -
- 4) Counting Sort

#### **[4] - Matrix in CPP –**

- 1) 2D Array Basic Introduction & Input, Output
- 2) Spiral matrix -
- 3) Diagonal Sum of Matrix -
- 4) Search for targeted element in Sorted Matrix -
  - 4.1) - Brute Force Approach - most simple method. T.C -  $O(n*m)$
  - 4.2) - Using Binary Search - 2nd Optimum - Row Wise -  $O(n*logn)$ , Colm Wise -  $O(m*logn)$
  - 4.3) - Staircase Search technique -  $O(n+m)$

#### **[5] - Character Array & Strings in CPP –**

- 1) Intro to Char Array –
- 2) Covert to Upper Letter -
- 3) Reverse a Char Array -
- 4) Valid Palindrome in char array -
- 5) Char Array(String) Functions -
- 6) String Intro & Basic I/p, o/p operations - 6.1) String Member Functions-
- 7) Check the two strings are Anagram or not -

## **[6] - STL - Vector**

1) Pair\_Sum - Find if any pair in Sorted Array has target sum

1.1) Brute Force Approach - using Nested Loops. TC -  $O(n^2)$

1.3) Using Linear Approach - 2 Pointer Approach. TC -  $O(n)$

## **[7] Bit Manipulations –**

1) Bitwise Operators -

2) Check if even or odd -

3) Get ith Bit of a num -

4) Set ith Bit - (sets the value to 1) -

5) Clr ith Bit - (sets the value to 0)

6) No. is power of 2 -

7) Count of Set Bits -

8) Fast Exponentiation -

## **7) Bit Manipulations –**

## **[8] OOP's in CPP –**

1) Classes & Object basic Intro -

1.1) Access Modifiers

1.2) Getters & Setters -

2) Encapsulations -

2.1) This Constructor

2.2) Types of Constructors -

2.3) Copy Constructor Concept -

2.4) Shallow & Deep Copy -

3) Destructor -

4) Inheritance -

4.1) Mode of Inheritance -

4.2) types of Inheritance -

4.2.1) Single Inheritance -

4.2.2) Multi-Level Inheritance -

4.2.3) Multiple Inheritance -

4.2.4) Hierarchical Inheritance -

4.2.5) Hybrid Inheritance -

5) Polymorphism -

5.1) Compiletime Polymorphism -

5.1.1) Function Overloading -

5.1.2) Function Overloading -

5.2) Polymorphism -

5.2.1) Function Overriding -

5.2.2) Virtual Functions -

6) Abstraction -

7) Static Keyword -

8) Friend Class &

Friend Function -

## **[9] Recursion –**

- 1)Recursion Basic Understanding-
- 2)Recursion Qn's -
  - 2.1) Print the numbers in descending Order
  - 2.2) Stack Overflow -
  - 2.3) Sum of N-Natural Numbers -
  - 2.4) Print Nth Fibonacci Number -
  - 2.5) Check if Array is Sorted or Not -
  - 2.6) First Occurrence -
  - 2.6.1) Last Occurrence -
  - 2.7) X to the power N -
- 3) Tiling problem
- 4) Remove Duplicates from the String -
- 5) Friends Pairing Problem -
- 6) Binary String Problem –

## **[10] Divide & Conquer –**

- 1) Merge Sort Algo -
- 2) Quick Sort Algorithm -
  - 2.1) Quick Sort Worst Case Scenario -
- 3) Searching In Rotated Array Problem -

## **Into to programming -**

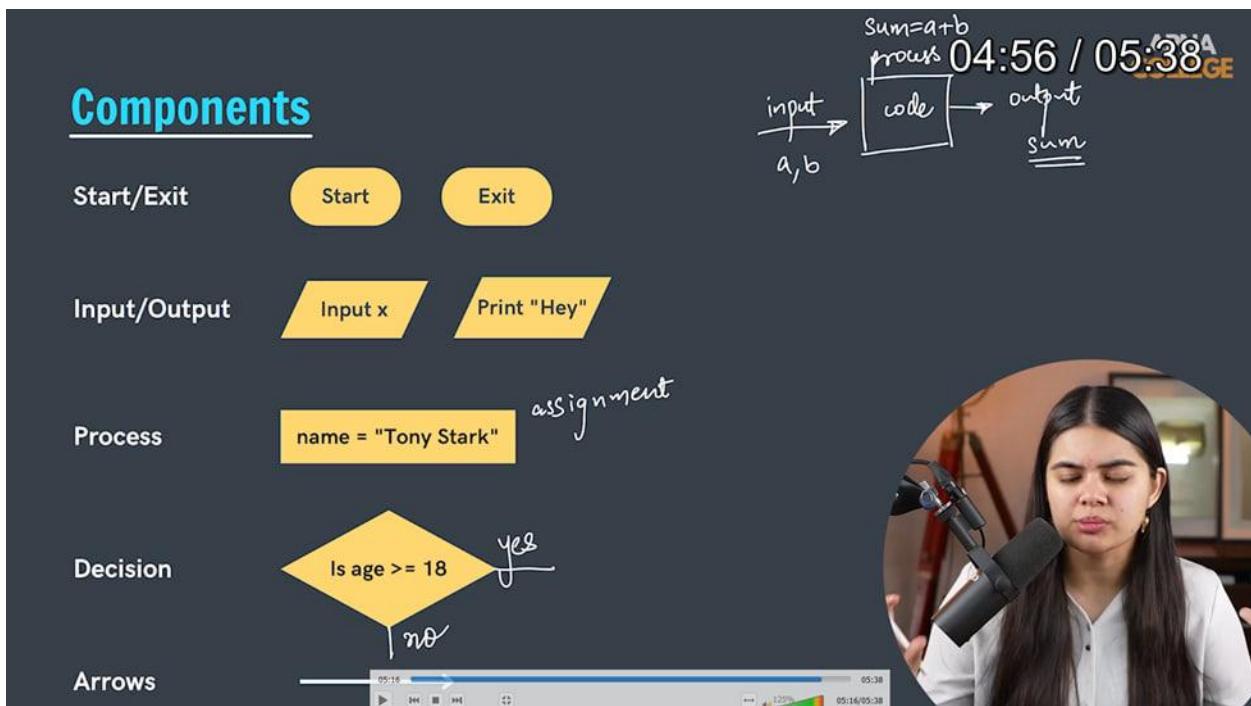
- 1) Basic Into to programming and Fundamentals - Flow Charts & Pseudo Codes ,Conditional & Looping Statement, Break & Continue, Functions, Pointers
- 2) digit sum -
- 3) Reverse Number -
- 4) Prime no. -
- 5) Nested Loops & Patterns -
  - 5.2) Star Pattern
  - 5.3) Inverted Star Pattern
  - 5.4) half Pyramid Pattern
  - 5.5) Character Pyramid Pattern
  - 5.6) Hollow Rectangle Pattern
  - 5.7) Inverted & Rotated Half Pyramid
  - 5.8) Floyd's Triangle
  - 5.9) Diamond Pattern
- 5.10) Butterfly Pattern
- 6) Number System -
  - 6.1) Binary to Decimal Conversion -

## 6.2) Decimal to Binary -

### 1\_IntrotoProgramming –

#### 1) Basic Into to programming and Fundamentals

##### Flow Charts & Pseudo Codes –



## Sum of 2 Numbers

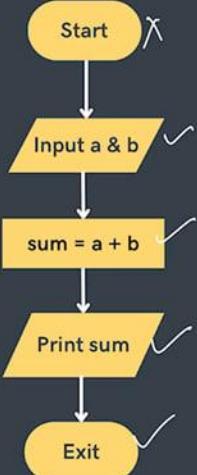
### Input

2 numbers - a & b

### Output

sum of a & b

APNA  
COLLEGE



### Pseudocode

1. Input a & b
2. sum = a + b
3. Print sum
4. Exit



## Calculate Simple Interest

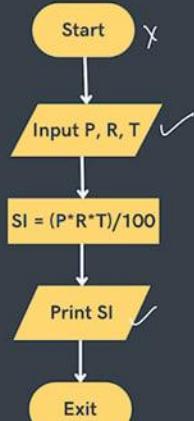
### Input

Principal, P  
Rate, R  
Time, T

### Output

SI = P\*R\*T/100

APNA  
COLLEGE



### Pseudocode

1. Input P, R, T
2.  $SI = (P * R * T) / 100$
3. Print SI
4. Exit



## Print max of 2 numbers

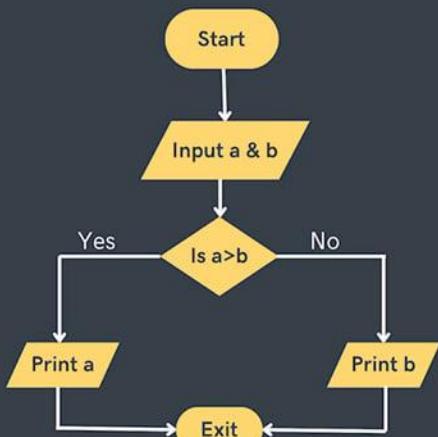
Input

a & b

Output

Larger of a & b

APNA  
COLLEGE



Pseudocode

1. Input a & b
2. If  $a \geq b$  yes  
    Print a  
Else  
    Print b
3. Exit

If —

Else



05:22 05:27 120% 05:22/05:27

## Print first N natural numbers

Day Run N=4 APNA  
COLLEGE

i=1

N=4.

1 <= 4 → yes

i=2

2 <= 4 → yes

i=3

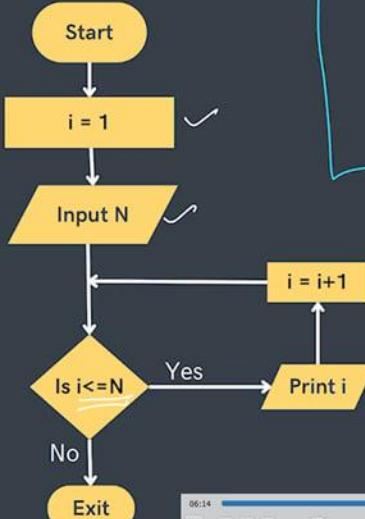
3 <= 4 → yes

i=4

4 <= 4 → yes

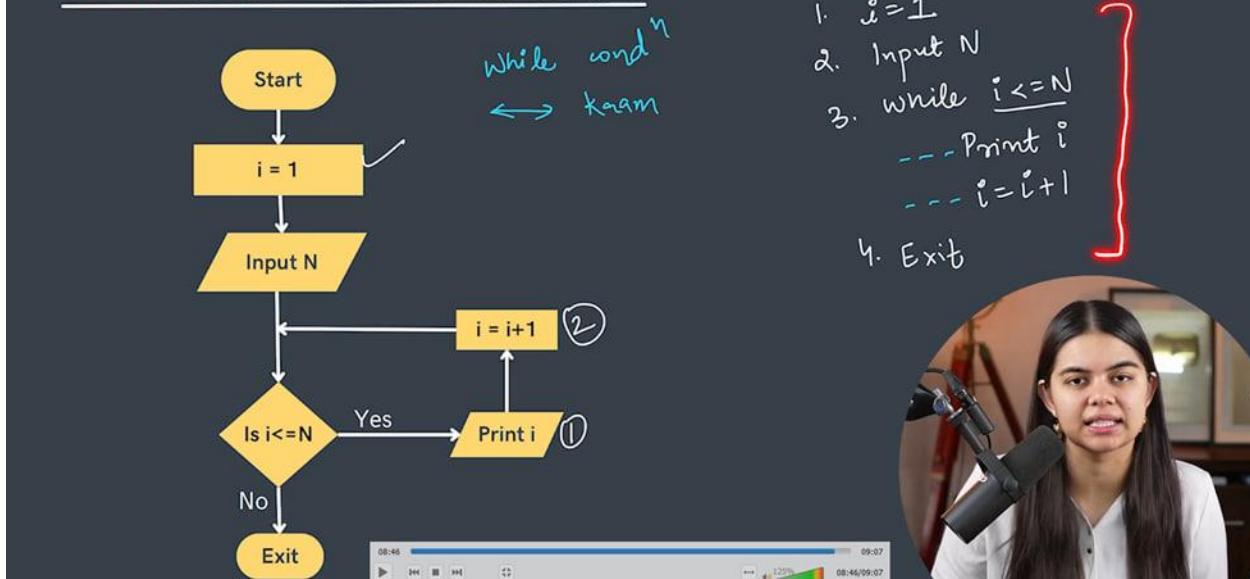
i=5

5 <= 4 → no  
exit

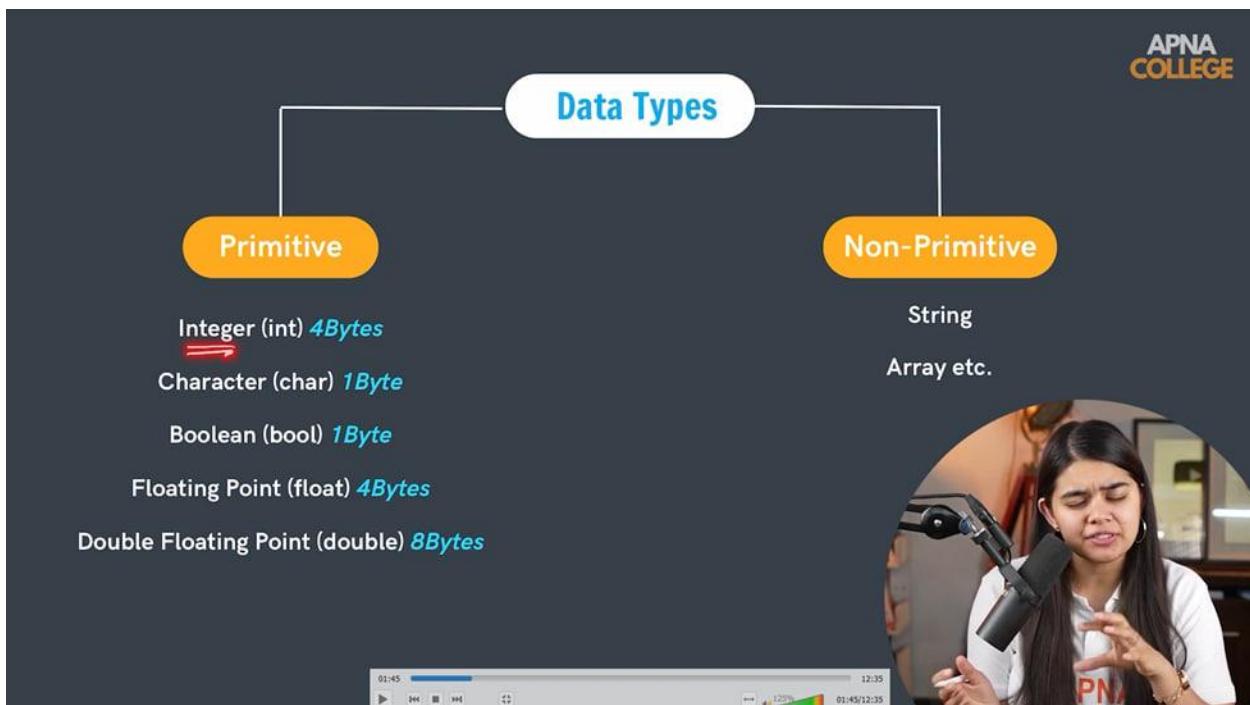


06:14 09:07 120% 06:14/09:07

## Print first N natural numbers



## Datatypes in cpp –



```
code.cpp
code.cpp > main()
5     int age = 25;
6     int marks = -200;
7     char grade = 'A';
8     bool isAdult = true;
9
10    cout<<"size of int = "<<sizeof(int)<<endl;
11    cout<<"size of char = "<<sizeof(char)<<endl;
12    cout<<"size of bool = "<<sizeof(bool)<<endl;
13
14 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh +

- apnacollege@Amans-MacBook-Pro Cpp Codes % g++ code.cpp
- apnacollege@Amans-MacBook-Pro Cpp Codes % ./a.out

```
size of int = 4
size of char = 1
size of bool = 1
```

- apnacollege@Amans-MacBook-Pro Cpp Codes %

Ln 13, Col 14 Spaces: 4 UTF-8

## Constants in cpp –

01. Constants.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Constants

Variables that cannot be changed.

```
const int n = 25; //must be initialized
```

const float PI = 3.14;

//not allowed

```
const int n;
```

n = 25;

macro → mem space X  
const → ✓ ✓ ✓

PI } const

APNA COLLEGE

## Typecasting in cpp -

The slide has a dark background. At the top left is the title "Typecasting". Below it is the subtitle "Conversion of data from one type to another.". To the right is a handwritten diagram showing the conversion of characters to integers: "'A' → 65, 'B' → 66, 'C' → 67, followed by three dots. Below this is another set of conversions: "'a' → 97, 'b' → 98, 'c' → 99, also followed by three dots. In the bottom left corner, there is a code snippet in a code editor:

```
int main() {
    cout << (10/3); //3
    cout << (10/3.0); //3.3333
    return 0;
}
```

Below the code, the text "bool -> char -> int -> float -> double" is displayed. On the right side of the slide, there is a circular video frame showing a woman with long dark hair speaking into a microphone.

## Explicit Typecasting -

The terminal window shows the following code in "code.cpp":

```
#include <iostream>
using namespace std;

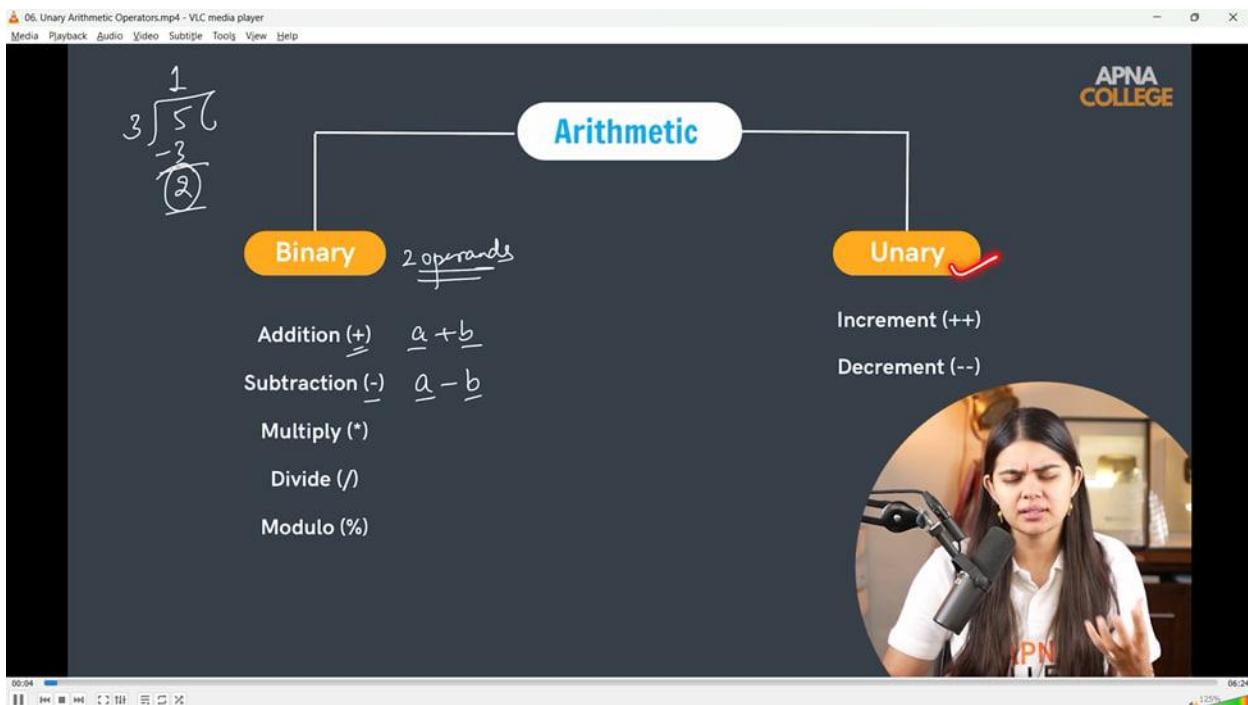
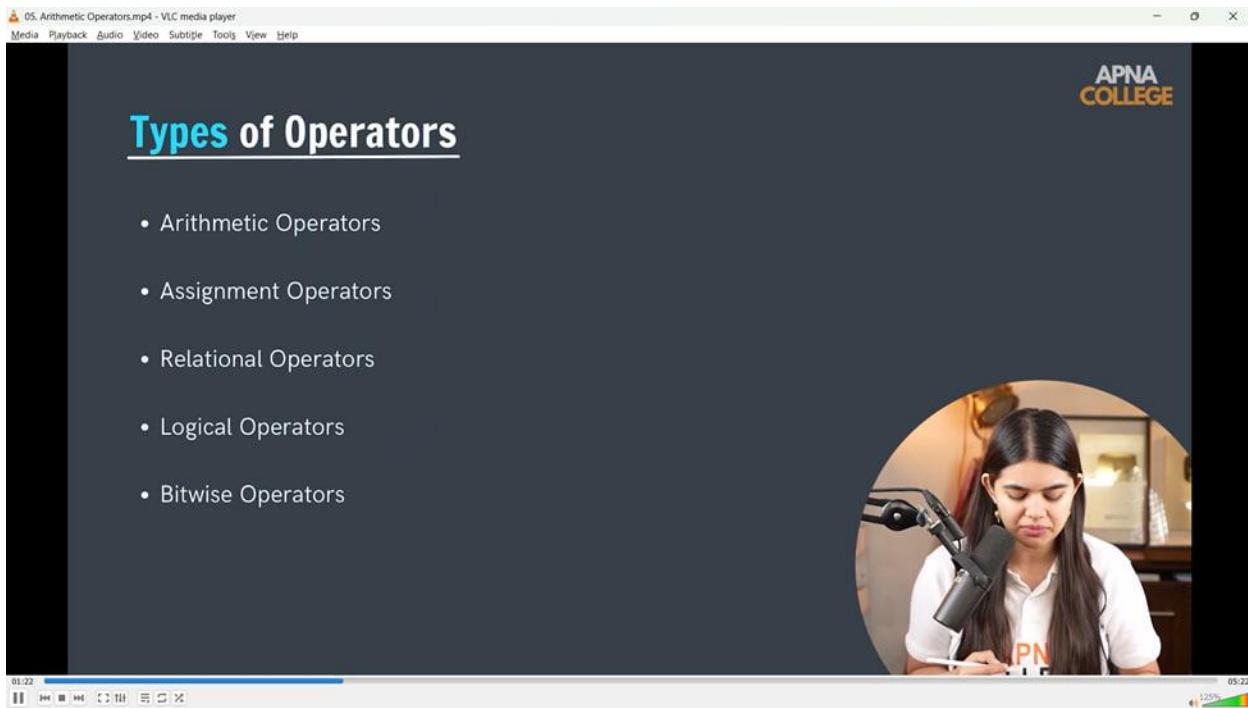
int main() {
    float PI = 3.14;
    cout << (int)(PI) << endl;
    cout << ((float)10/3) << endl; //3.3333
    cout << (char)['A' + 3] << endl; //B
    return 0;
}
```

The terminal output shows the results of the typecasted values:

```
3
3.33333
3
3.33333
B
```

On the right side of the terminal window, there is a circular video frame showing the same woman speaking into a microphone.

## Operatoes in cpp –



## Increment Decrement Operator -



A screenshot of a video player window titled "07. Assignment Operators.mp4 - VLC media player". The video frame shows a woman with long dark hair, wearing a white shirt, speaking into a black microphone. The VLC interface includes a toolbar at the top with icons for file operations, a search bar, and the APNA COLLEGE logo. Below the video frame is a terminal window showing the output of a C++ program. The code in the editor is:

```
13 // cout << "% " << (a % b) << endl; //Modulo (remainder) : 2
14
15 int a = 3;
16 int b = a++;
17 cout << "a = " << a << endl; //4
18 cout << "b = " << b << endl; //3
19 return 0;
20 }
```

The terminal output shows:

```
a = 4
b = 4
● apnacollege@Amans-MacBook-Pro Cpp Codes % g++ code.cpp
● apnacollege@Amans-MacBook-Pro Cpp Codes % ./a.out
a = 4
b = 3
○ apnacollege@Amans-MacBook-Pro Cpp Codes %
```

## Assignment Operator –



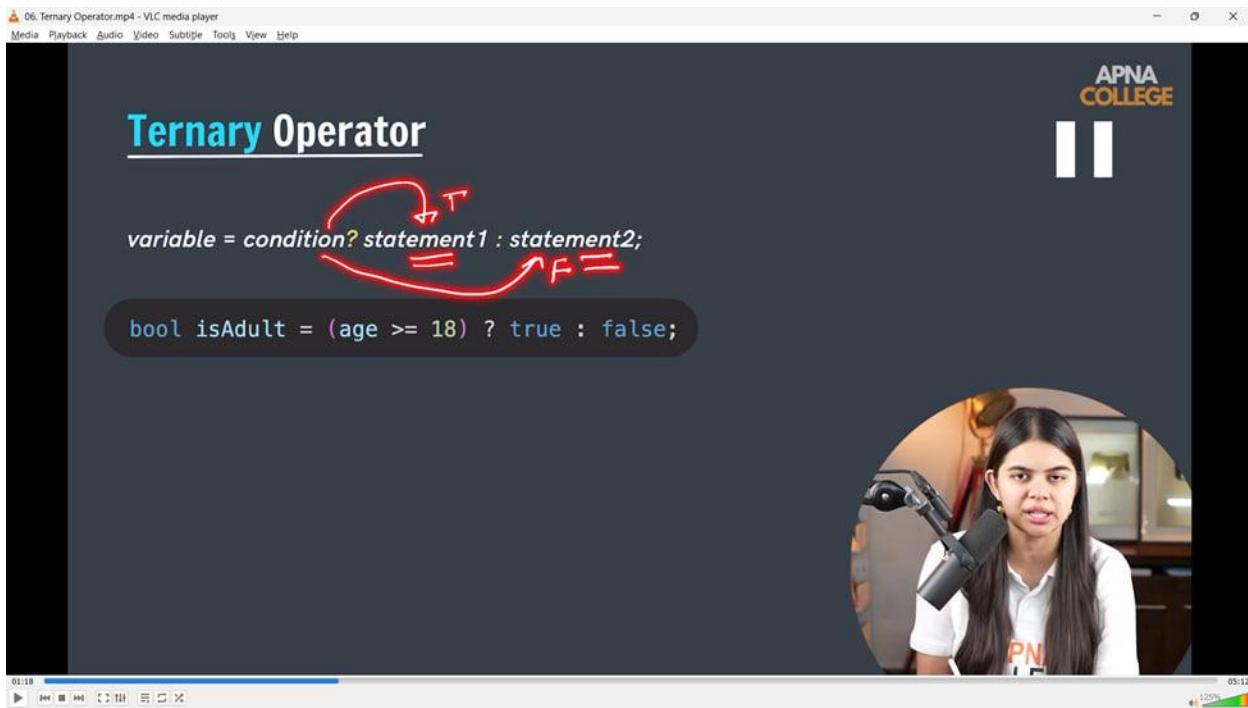
A screenshot of a video player window titled "07. Assignment Operators.mp4 - VLC media player". The video frame shows a woman with long dark hair, wearing a white shirt, speaking into a black microphone. The VLC interface includes a toolbar at the top with icons for file operations, a search bar, and the APNA COLLEGE logo. Below the video frame is a terminal window showing the output of a C++ program. The code in the editor is:

```
18 // cout << "b = " << b << endl; //2
19
20 //Assignment Operators
21 int a = 10;
22
23 a += 5; // a = a + 5 => 15
24 cout << a << endl; //15
25 a -= 5; // a = a- 5 => 5
26 cout << a << endl; //5
27 a *= 5;
28 cout << a << endl; //50
29 a /= 5;
30 cout << a << endl; //2
31
32 return 0;
```

The terminal output shows:

```
● apnacollege@Amans-MacBook-Pro Cpp Codes % g++ code.cpp && ./a.out
15
10
50
10
○ apnacollege@Amans-MacBook-Pro Cpp Codes %
```

## Ternary Operator –



## //1) Basic Into to programming and Fundamentals

```
int main()
{
    float PI = 3.14159265359;
    double PI2 = 3.14159265359;
```

```
cout << "PI = " << PI << endl;
```

```
cout << "PI 2 = " << PI2 << endl;
```

```
// PI = 3.14159 - by defaalut the precision is 5. for increment it we can use setprecision  
function
```

```
// PI 2 = 3.14159

// setprecision function
cout << setprecision(10) << "PI = " << PI << endl;
cout << setprecision(10) << "PI2 = " << PI2 << endl;

// PI = 3.141592741
// PI2 = 3.141592654

// Constants -
const int n = 25; // must be initialized while defining
const float py = 3.14;

cout << "n is - " << n << endl;
cout << "py is - " << py << endl;

cout << "g - \n"
    << g; // g - 9.8

// Typecasting order when different datatypes -
// bool -> char -> int -> float -> double

// Operators -
// ternary Op - simply single line code instead of if else
bool isadult; // 1,0
int age;
```

```
cout << "What is the age " << endl;  
// cin >> age;
```

```
isadult = age >= 18 ? true : false;  
cout << isadult << endl; // 1
```

```
// largests of 2 mnumbers =-  
int a = 5;  
int b = 10;  
int largest = a >= b ? a : b;  
cout << largest << endl; // 10
```

```
// Loops in cpp -  
// basoc square star pattern -  
for (int i = 1; i <= 4; i++)  
{  
    cout << "****" << endl;
```

```
/*  
****  
****  
****  
****  
*/  
for (int i = 5; i > 0; i--)
```

```

{
    cout << i << "\n";
}
// _____
```

**Reverse the number problem –**

## Practice Qs

Qs. Print the digits of a given number in reverse using **while** loop.

$n = 12345$

$$\begin{array}{l} \text{last Dig} = n \% 10 \\ \text{res} = res * 10 + \text{last Dig} \\ n = n / 10 \end{array}$$

$\text{res} = res * 10 + \text{last Dig}$



$$\begin{array}{r}
 523 \\
 \downarrow \\
 5 \times 10^2 = 500 \\
 + 2 \times 10^1 = 20 \\
 + 3 \times 10^0 = 3 \\
 \hline
 523
 \end{array}$$

Qs. Reverse a given number & print the result.

$res = 0$	$0 * 10 + 9 = 9$
$res = 9$	$9 * 10 + 2 = 92$
$res = 92$	$92 * 10 + 8 = 928$
$res = 928$	$928 * 10 + 0 = 9280$
$res = 9280$	$9280 * 10 + 1 = 9281$



**//2) digit sum -**

```

// int m;
// cout << "Enter number value - " << endl;
// cin >> m;
```

```
// int sumdigit=0;
```

```

// while (m > 0)
//{
//    int lastdigit = m % 10;
```

```

//    sumdigit += lastdigit;
//    m /= 10;
// }

// cout<<"So, the digit sum is - "<<sumdigit<<endl;

// Odd digit sum -
// int m;
// cout << "Enter number value - " << endl;
// cin >> m;

// int sumdigit = 0;

// while (m > 0)
// {
//     int lastdigit = m % 10;
//     if (lastdigit % 2 != 0)
//     {
//         sumdigit += lastdigit;
//     }

//     m /= 10;
// }

// cout << "So, the digit sum is - " << sumdigit << endl;
// _____
//3) Reverse Number -
// int m;

```

```

// cout << "M's value - " << endl;
// cin >> m;
// int revesenumber;

// while (m > 0)

//{
//    revesenumber = m % 10;
//    cout<<revesenumber<<" ";
//    m /= 10;
//}

// _____

```

### Do-while loop –

**do-while Loop**

01:22 / 04:15  
do-while Loop  
MCA COLLEGE

```

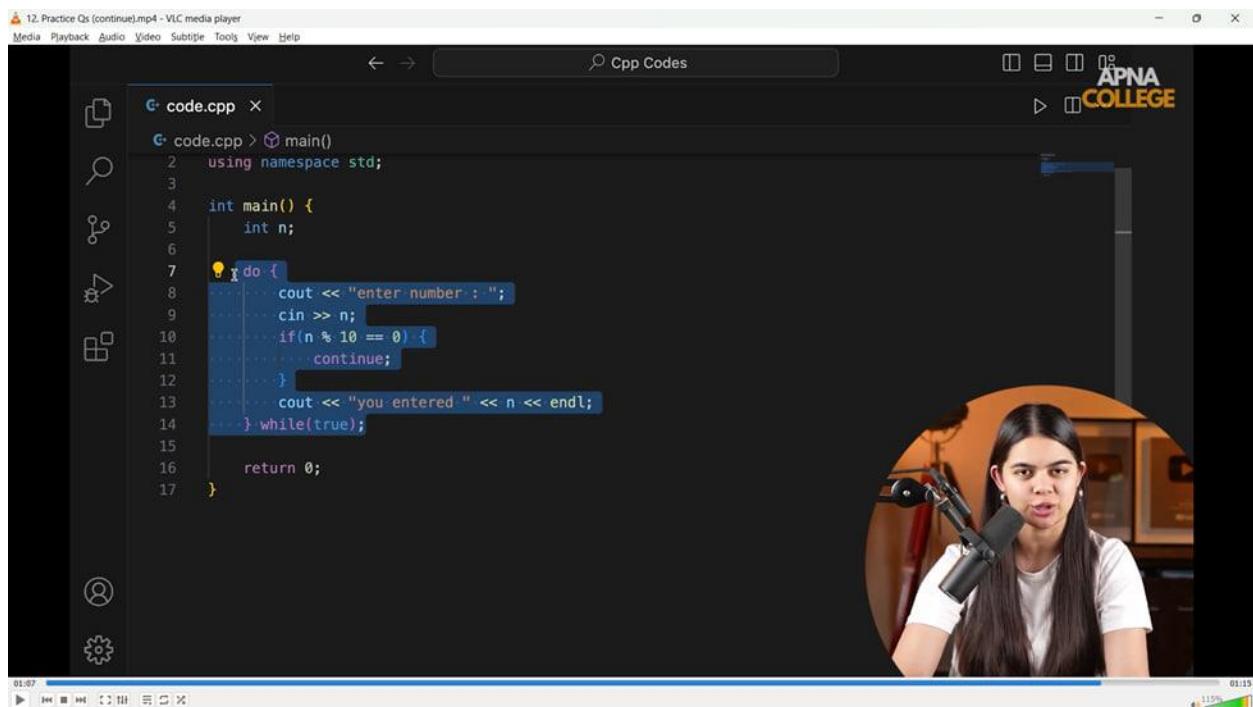
do {
    //do some work
} while(condition);

```

//work gets done atleast once irrespective of condition

The video player interface includes a timestamp at the top right (01:22 / 04:15), the video title 'do-while Loop' in blue, and the channel name 'MCA COLLEGE' in orange. The video content itself shows a person speaking into a black microphone, with handwritten notes overlaid on the screen illustrating the do-while loop structure.

## Continue -



// Break Statement -

// do-while loop basic syntax -

```
// int val = 1;  
// do  
// {  
//   cout << "Learning do-while loop here " << endl;  
// } while (val > 5);  
// // Learning do-while loop here
```

// while (val > 5)

```
// {
```

```
// cout << "Learning using while loop here" << endl;
// }

/// Break statement in looping -

// int i = 1;

// while (i <= 10)

//{
// if (i == 3)
// {
//     break;
// }

// cout << i << endl;
// i++;
//}

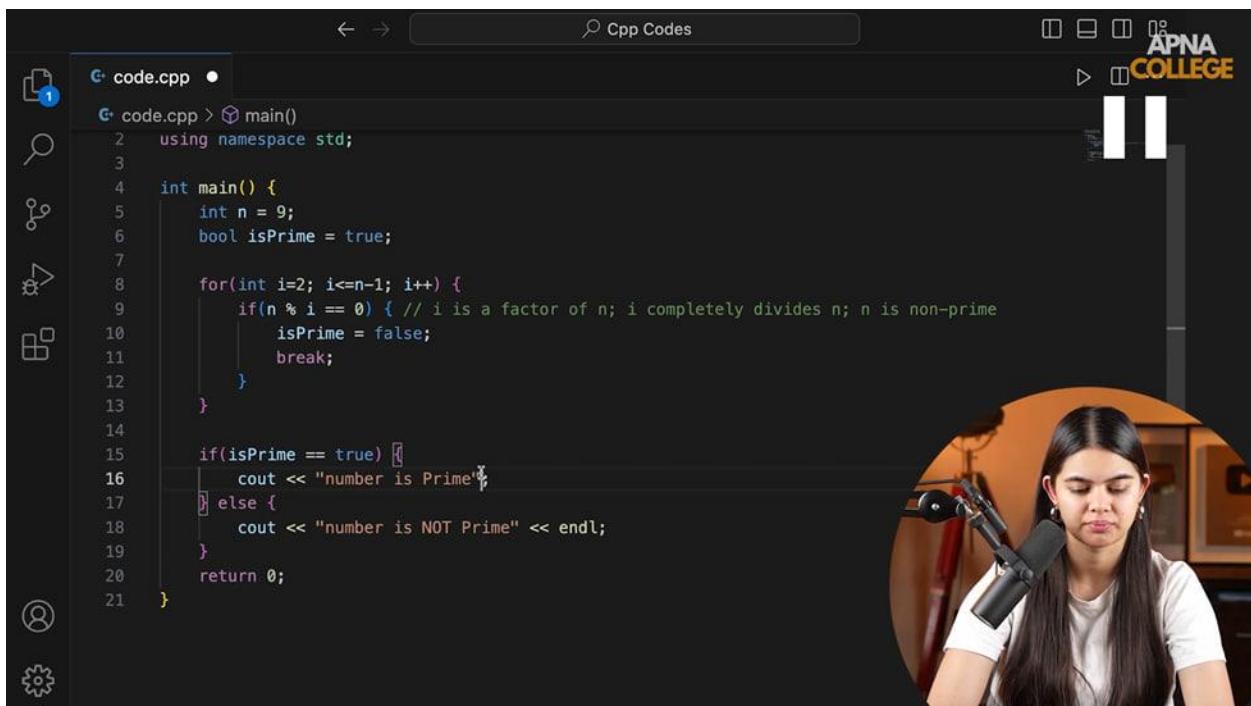
/*
1
2
*/



// int n;
// do {
//     cout << "Enter Number - ";
//     cin >> n;
//     if (n % 10 == 0) {
//         break;
//     }
}
```

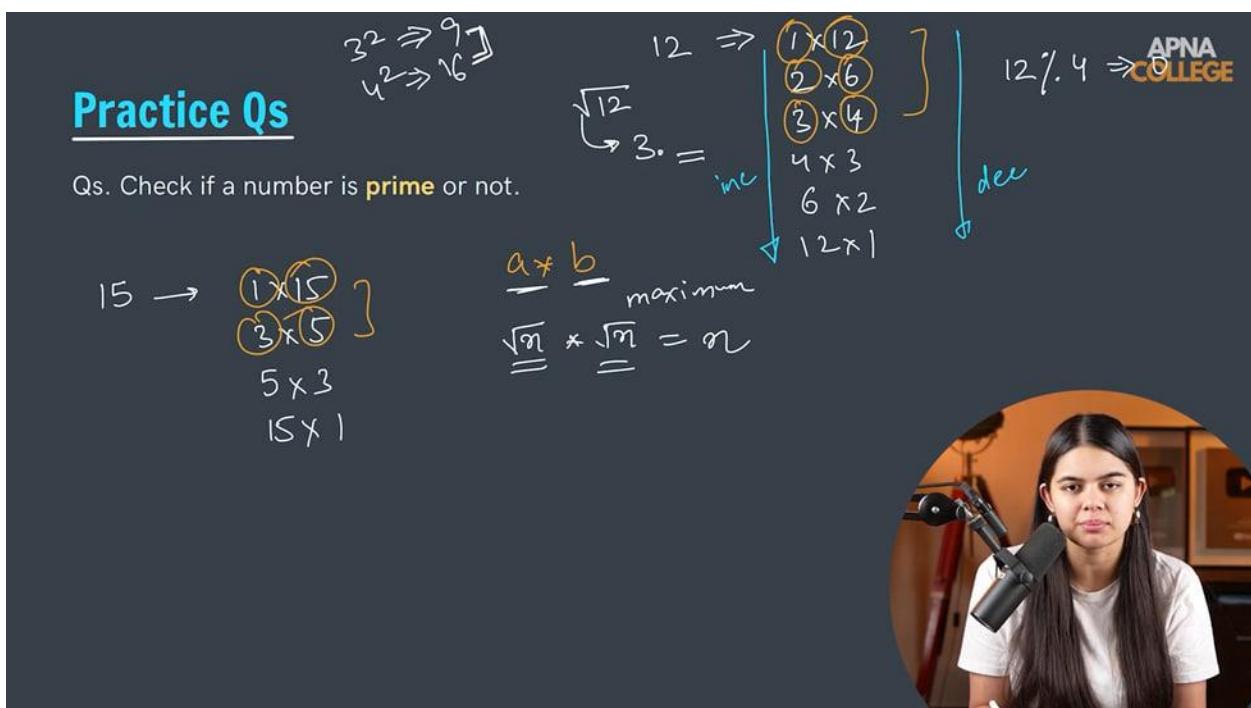
```
// cout << "You entered here - " << n << endl;  
// } while (true);  
  
// return 0;  
// _____  
// Continue statement -  
  
// for (int i = 1; i <= 10; i++)  
// {  
//   if (i == 3)  
//   {  
//     continue;  
//   }  
//   cout << i << endl;  
// }  
// /*  
// 1  
// 2  
// 4  
// 5  
// 6  
// 7  
// 8  
// 9  
// 10  
// */  
// _____
```

#### //4) Prime no. -



```
code.cpp
1 using namespace std;
2
3 int main() {
4     int n = 9;
5     bool isPrime = true;
6
7     for(int i=2; i<=n-1; i++) {
8         if(n % i == 0) { // i is a factor of n; i completely divides n; n is non-prime
9             isPrime = false;
10            break;
11        }
12    }
13
14    if(isPrime == true) {
15        cout << "number is Prime";
16    } else {
17        cout << "number is NOT Prime" << endl;
18    }
19
20    return 0;
21 }
```

#### Prime Nuo. Squareroot logic -



**Practice Qs**

Qs. Check if a number is **prime** or not.

$3^2 \Rightarrow 9$     $\sqrt{9} \Rightarrow 3$

$4^2 \Rightarrow 16$     $\sqrt{16} \Rightarrow 4$

$\sqrt{12} \Rightarrow 3$     $12 \Rightarrow [1 \times 12, 2 \times 6, 3 \times 4]$

$12 \div 4 \Rightarrow \text{Ans}$

$15 \rightarrow [1 \times 15, 3 \times 5]$

$a * b = \text{maximum } \sqrt{n} * \sqrt{n} = n$

## Practice Qs

Qs. Check if a number is **prime** or not.

$$21 \Rightarrow 1 \times 21 \\ 3 \times 7 \\ 7 \times 3 \\ 21 \nmid 1$$

$$\sqrt{21} \Rightarrow 4. -$$

$$2, 3, 4 \\ 2 \leq \sqrt{n}$$



```
// int num;

// cout << "Enter the no. you wants to check for Prime or Not" << endl;
// cin >> num;

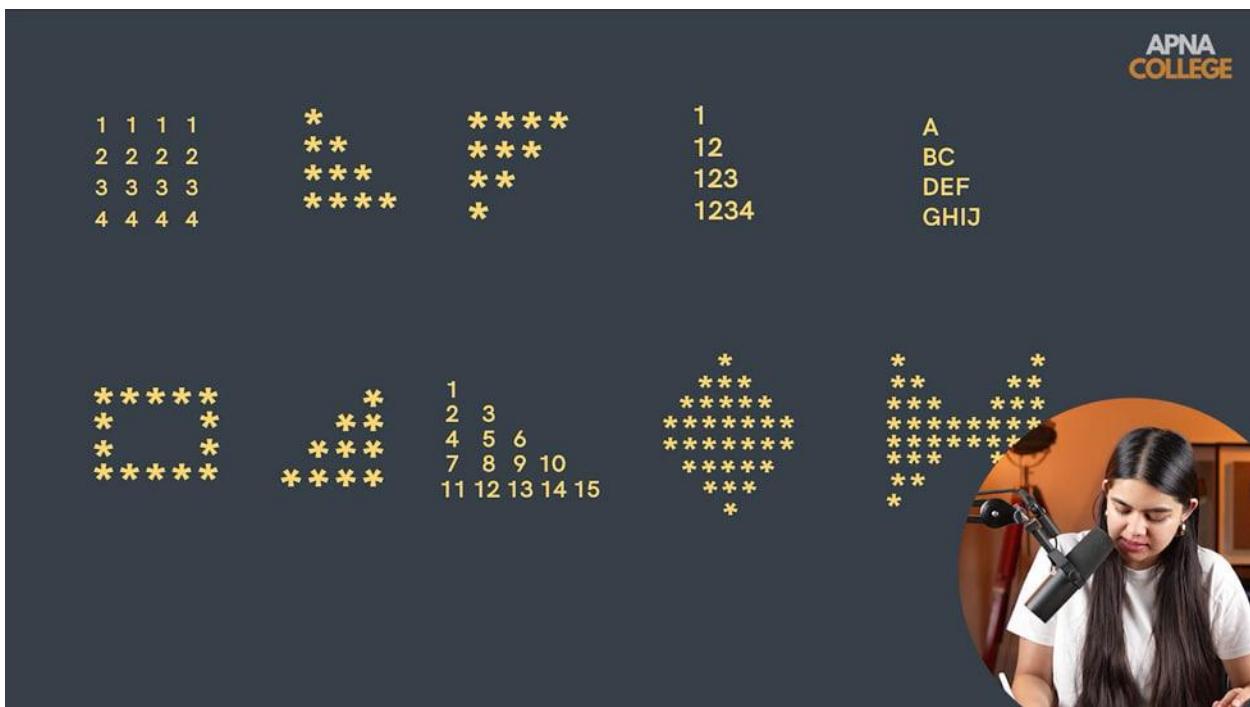
// if (num == 1)
// {
//   cout << "No. is Neither Prime Nor Composite" << endl;
// }

// for (int i = 2; i <= sqrt(num); i++)
// {
//   if (num % i == 0)
//   {
//     cout << "Not a Prime Number" << endl;
//   }
//   return 0;
```

```
// }  
// }  
// cout << "prime number" << endl;
```

```
//
```

**Patterns in cpp –**



**//5) Nested Loops & Patterns –**

```
/*
```

**// 1) Print the Nested Loop Pattern -**

```
// 1 1 1 1
```

```
// 2 2 2 2
```

```
// 3 3 3 3
```

```
// 4 4 4 4
```

```
// */
```

```
// int n;
```

```
// cout << "n's value " << endl;  
// cin >> n;
```

```
// for (int i = 1; i <= n; i++)  
// {  
//   for (int j = 1; j <= n; j++)  
//   {  
//     cout << i << " ";  
//   }  
//   cout << endl;  
// }
```

```
// /* O/p -
```

```
// n's value  
// 6  
// 1 1 1 1 1 1  
// 2 2 2 2 2 2  
// 3 3 3 3 3 3  
// 4 4 4 4 4 4  
// 5 5 5 5 5 5  
// 6 6 6 6 6 6  
// */  
// _____
```

// Nested Loops -

**Nested Loops**

Loop inside a loop

Row 1 C1 C2 C3 C4  
R1 1 1 1 1  
R2 2 2 2 2  
R3 3 3 3 3  
R4 4 4 4 4

① Outer loop : no. of Rows  
② Inner loop : no. of columns / each row  
③ Work in inner loop (each row)  
    i print

```
for(int i=1; i<=4; i++) {  
    for(int j=1; j<=4; j++) {  
        cout << i << " "  
    }  
}
```

APNA COLLEGE

## 5.2) Star Pattern

Print Star pattern

R1 \* 1st  
R2 \*\* 2nd  
R3 \*\*\* 3rd  
R4 \*\*\*\* 4th

m = 4

① outer loop → Rows (1) → Rows n times (1 to n)  
② inner loop (each row) i times (1 to i)  
③ work ?

n = 4

APNA COLLEGE

/\* 2) Print the Star Pattern

```
// *
```

```
// *
// ***
// ****
// *****

// */
// int n;
// cout << "vakue of n - " << endl;
// cin >> n;

// for (int i = 1; i <= n; i++)
// {
//     for (int j = 1; j <= i; j++)
//     {
//         cout << "* ";
//     }
//     cout << endl;
// }

// /*
// 5
// *
// **
// ***
// ****
```

```
// _____  
// /* 3) Print the Inverted Star Pattern  
  
// * * * * *  
// * * * *  
// * * *  
// * *  
// *  
// */  
// int n;  
// cout << "value of n - " << endl;  
// cin >> n;  
  
// for (int i = 1; i <= n; i++)  
// {  
//   for (int j = 1; j <= n - i + 1; j++)  
//   {  
//     cout << "* ";  
//   }  
//   cout << endl;  
// }  
// /*  
// value of n -  
// 5  
// * * * * *  
// * * * *  
// * * *
```

```
// * *
// *
// */
// _____
// /* 4) Print the half Pyramid Pattern

// 1
// 1 2
// 1 2 3
// 1 2 3 4
// 1 2 3 4 5
// */
// int n;
// cout << "vakue of n - " << endl;
// cin >> n;

// for (int i = 1; i <= n; i++)
// {
//     for (int j = 1; j <= i; j++)
//     {
//         cout << j << " ";
//     }
//     cout << endl;
// }

// /*
// vakue of n -
// 5
```

```
// 1  
// 1 2  
// 1 2 3  
// 1 2 3 4  
// 1 2 3 4 5  
// */
```

```
// _____  
// /*
```

#### // 4.1) Print the Pattern

```
// 1  
// 2 2  
// 3 3 3  
// 4 4 4 4  
// 5 5 5 5 5  
// */  
// int n;  
// cout << "vakue of n - " << endl;  
// cin >> n;
```

```
// for (int i = 1; i <= n; i++)  
// {  
//     for (int j = 1; j <= i; j++)  
//     {  
//         cout << i << " ";  
//     }  
// }
```

```
//      cout << endl;  
//  }  
// /*  
// value of n -  
// 5  
// 1  
// 2 2  
// 3 3 3  
// 4 4 4 4  
// 5 5 5 5 5  
// */
```

```
// _____  
// /*  
// 5) Print the Character Pyramid Pattern  
// A  
// B C  
// D E F  
// G H I J  
// K L M N O  
// */  
// int n;  
// cout << "value of n - " << endl;  
// cin >> n;  
// char ch = 'A';
```

```
//  for (int i = 1; i <= n; i++)  
//  {  
//      for (int j = 1; j <= i; j++)  
//      {  
//          cout << ch << " ";  
//          ch++;  
//      }  
//      cout << endl;  
//  }  
// /*  
//  value of n -  
//  5  
//  A  
//  B C  
//  D E F  
//  G H I J  
//  K L M N O  
//  */  
// _____  
// /*
```

### **// 5.1) Print the Character Pyramid Pattern for same Char in the line**

```
// A  
// B B  
// C C C  
// D D D D  
// E E E E E
```

```
// */
// int n;
// cout << "vakue of n - " << endl;
// cin >> n;
// char ch = 'A';

// for (int i = 1; i <= n; i++)
//{
//    for (int j = 1; j <= i; j++)
//    {
//        cout << ch << " ";
//    }
//    ch++;
//    cout << endl;
//}

// /*
// value of n -
// 5
// A
// B B
// C C C
// D D D D
// E E E E E
// */

// _____
```

```
/*
```

## 6) Print the Hollow Rectabgle Patternn

```
*****
```

```
* * *
```

```
* * *
```

```
* * *
```

```
*****
```

```
*/
```

① Outer loop (rows) (1 to n)

② Inner loop (each row)

```
cout << "*" ; //First
for( 1 to n-1 ) {
    1st or last -> "*"
    else -> " "
}
cout << "*" ; //last
```

```
// int n;
```

```
// cout << "vakuue of n - " << endl;
```

```
// cin >> n;
```

```
// for (int i = 1; i <= n; i++)
```

```
// {
```

```
// cout<<"*";//First Row
// for (int j = 1; j <= n - 1; j++)//sapce printing
// {
//     if (i == 1 || i == n)
//     {
//         cout << "*";
//     }
//     else
//     {
//         cout << " ";
//     }
// }
// cout<<"*"><<endl;//Last Row
// }
```

```
// Lgic 2
// for (int i = 1; i <= n; i++)
// {
//     for (int j = 1; j <= n; j++)
//     {
//         if (i == 1 || i == n || j == 1 || j == n)
//         {
//             cout << "* ";
//         }
//         else
//         {
```

```
//      cout<<" ";
//    }
//  }
//  cout << endl;
// }
```

```
/*
value of n -
```

5

```
* * * * *
*   *
*   *
*   *
* * * * *
*/
```

```
// _____
// /*
```

### **// 7) Print Inverted & Rotated Half Pyramid**

```
//   *
//   * *
//   * * *
//   * * * *
// * * * * *
// */
```

$n = 4$

## Inverted & Rotated Half-Pyramid

$n = 4$



$n = 4$

## Inverted & Rotated Half-Pyramid

$n = 4$

$i = 1$	$4 - 1 = 3$
$i = 2$	$4 - 2 = 2$
$i = 3$	$4 - 3 = 1$
$i = 4$	$4 - 4 = 0$



```
// int n;
// cout << "value of n - " << endl;
// cin >> n;
```

```
//   for (int i = 1; i <= n; i++)  
//   {  
//     // Printoing SPce  
//     for (int j = 1; j <= n - i; j++)  
//     {  
//       cout << " ";  
//     }  
//   }
```

```
//   // Printing Str  
//   for (int k = 1; k <= i; k++)  
//   {  
//     cout << "* ";  
//   }  
//   cout << endl;  
// }  
// /*  
// value of n -  
// 5  
// *  
// **  
// ***  
// ****  
// *****  
// */
```

```

// _____
// /*
// 8) Print Floyd's Triangle
// 1
// 2 3
// 4 5 6
// 7 8 9 10
// 11 12 13 14 15
// */

```

**Print Floyd's Triangle**

num=1

① outer loop (rows)  
(1 to n)

② inner loop (each row)  
elements  
(1 to i)

③ work?  
 $\text{cout} \ll \text{num};$   
 $\text{num}++;$

R1 1  
R2 2 3  
R3 4 5 6  
R4 7 8 9 10  
R5 11 12 13 14 15

1st      1st  
2nd      2nd  
3rd      3rd  
4th      4th  
5th      5th

i times  
(1 to i)



APNA COLLEGE

```

// int n;
// cout << "value of n - " << endl;
// cin >> n;
// int num = 1;

```

```
// for (int i = 1; i <= n; i++)  
// {  
//   for (int j = 1; j <= i; j++)  
//   {  
//     cout << num << " ";  
//     num++;  
//   }  
//   cout << endl;  
// }
```

```
// /*  
// value of n -  
// 5  
// 1  
// 2 3  
// 4 5 6  
// 7 8 9 10  
// 11 12 13 14 15  
// */
```

```
// _____
```

```
// /*  
// 9) Print the Diamond Pattern  
//      *  
//      * * *  
//      * * * * *  
//      * * * * * * *  
//      * * * * * * * *  
//      * * * * * * * *  
//      * * * * * * *  
//      * * * * *  
//      * * *  
//      *  
//      */  
// int n;  
// cout << "vakuue of n - " << endl;  
// cin >> n;
```

```
// // 1st Pyramid  
// for (int i = 1; i <= n; i++)  
// {  
//     // Printing SSpace  
//     for (int j = 1; j <= n - i; j++)  
//     {  
//         cout << " ";
```

```
//      }

//      // Printing Star
//      for (int k = 1; k <= 2 * i - 1; k++)
//      {
//          cout << "* ";
//      }
//      cout << endl;
//  }

//  // 2nd Pyramid
//  for (int i = n; i >= 1; i--)
//  {
//      // Printing SSpace
//      for (int k = 1; k <= n - i; k++)
//      {
//          cout << " ";
//      }
//      // Printing Star
//      for (int j = 1; j <= 2 * i - 1; j++)
//      {
//          cout << "* ";
//      }
//      cout << endl;
//  }

// /*
```

```
// value of n -  
// 5  
//      *  
//      * * *  
//      * * * * *  
//      * * * * * * *  
//      * * * * * * * *  
//      * * * * * * * *  
//      * * * * * * *  
//      * * *  
//      *  
//      */  
// _____  
/*
```

### 10) Print the Butterfly Pattern

```
*          *  
* *        * *  
* * *      * * *  
* * * *    * * * *  
* * * * *  * * * *  
* * * * * *  * * *  
* * * * * * *  * *  
* * * * * * * *  *  
* * * * * * * * *  *  
* * * * * * * * * *  *
```

\*/

$n=4$

## Print Butterfly Pattern

$n = 4$

**Pattern1**

(1) outer loop (rows) (1 to n)

(2) inner loop

a) stars

b) spaces

c) stars

APNA COLLEGE

$R1 : 1st + 6sp + 1st$

$R2 : 2st + 4sp + 2st$

$R3 : 3st + 2sp + 3st$

$R4 : 4st + 0sp + 4st$

```
int n;  
cout << "value of n - " << endl;  
cin >> n;
```

```
// 1st Butterfly  
  
// Printing Star  
  
for (int i = 1; i <= n; i++)  
{  
    for (int j = 1; j <= i; j++)  
    {  
        cout << "* ";  
    }  
  
    // Printing Space
```

```
for (int j = 1; j <= 2 * (n - i); j++)  
{  
    cout << " ";
```

```
}  
  
// Printing Start for last  
for (int k = 1; k <= i; k++)  
{  
    cout << "* ";  
}  
cout << endl;  
}
```

```
// 2nd Half Butterfly  
for (int i = n; i >= 1; i--)  
{  
    for (int j = 1; j <= i; j++)  
    {  
        cout << "* ";  
    }
```

```
// Printing Space  
for (int j = 1; j <= 2 * (n - i); j++)  
{  
    cout << " ";
```

```
// Prinmting Start for last

for (int k = 1; k <= i; k++)
{
    cout << "* ";
}

cout << endl;

}

/*
value of n -
5

*      *
* *      * *
* * *      * * *
* * * *      * * * *
* * * * *      * * * *
* * * * * *      * *
* * * *      * *
* *      * *
*      *

*/
// _____
```

## Functions in cpp –

01. What are Functions-.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Functions

Block of code which runs when it is called.

```
returnType fName() {  
    //do some work  
    return someValue; //optional  
}  
  
fName(); //function call
```

void myFunction

APNA COLLEGE



02:38 09:03 125%

02. Forward Declaration.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Forward Declaration

Declaration: the function's name, return type, and parameters (if any)

Definition: the body of the function

→ int a; // declare  
 a=10; // assign  
 int a=10;

Declaration

Definition

```
void sayHello(){  
    cout << "Hello :)" \n;  
}
```

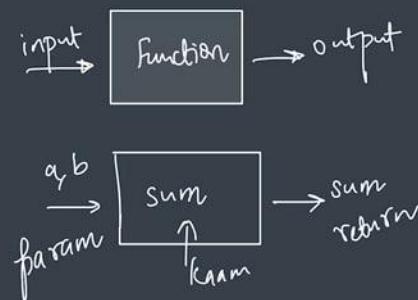
APNA COLLEGE



00:46 03:31 125%

## Syntax with Parameters

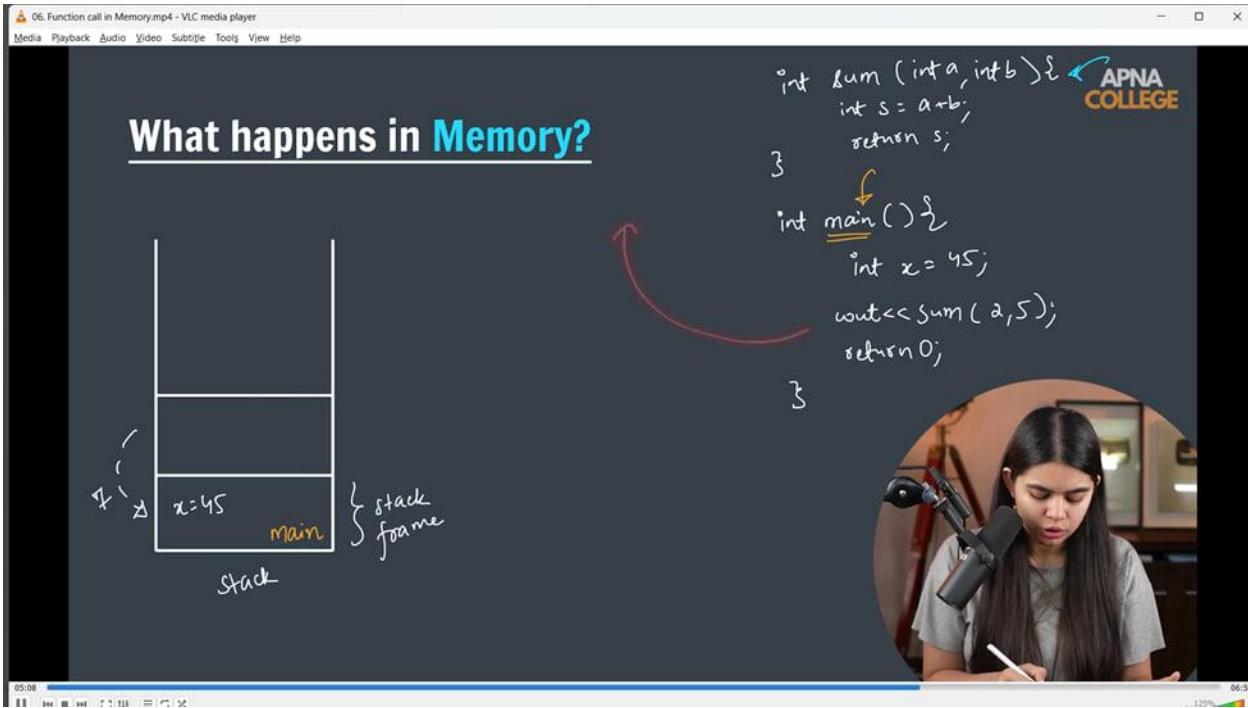
```
returnType fname ( type param1, type param2 ..) {  
    //do some work  
    return someValue;  
}
```



**Parameters – Function declare krne ke baad usme jo variables pass kiya jaate h is called Parameter.**

**Arguments – Function caaling me jo value pass hoti he use Arguments.**

**Forward Declaration -** When a function is declared before main function and defined after the main function. It's forward Declaration. Pehele hi declare kr diya and ab define kr rheh



```
// void sayHello() // Function Declaration
// {
//     // inside the function its Funciton Definition - How is the function defined to do
//     work
//
//     cout << "Hello, Shubham you'll be get placed in Microsoft in August 2025" << endl;
//
//     cout << "All the very best." << endl;
//
// }
//
// void assistance()
//
// {
//
//     sayHello();
//
//     cout << "Work done boss" << endl;
//
// }
```

```
// int sum(int a, int b){ // a & b are Parameters here.
//
//     int sum = a + b;
//
//     return sum;
```

```
// }

// int mul(int a, int b){
//   int mul = a*b;
//   return mul;
// }

// int sub(int a, int b=1){ //passing a value in 2nd parameter as by default 1, if argumeent didn't
// pass in function then system will take this 1 as 2nd arguement
//   int sub = a-b;
// }

// int main()
//{
//   sayHello();
//   sayHello(); // Function Calling
//   sayHello();

//   assistance();
//   /*
//   Hello, Shubham you'll be get placed in Microsoft in August 2025
//   All the very best.

//   Hello, Shubham you'll be get placed in Microsoft in August 2025
//   All the very best.

//   Hello, Shubham you'll be get placed in Microsoft in August 2025
//   All the very best.

//   Hello, Shubham you'll be get placed in Microsoft in August 2025
//   All the very best.
```

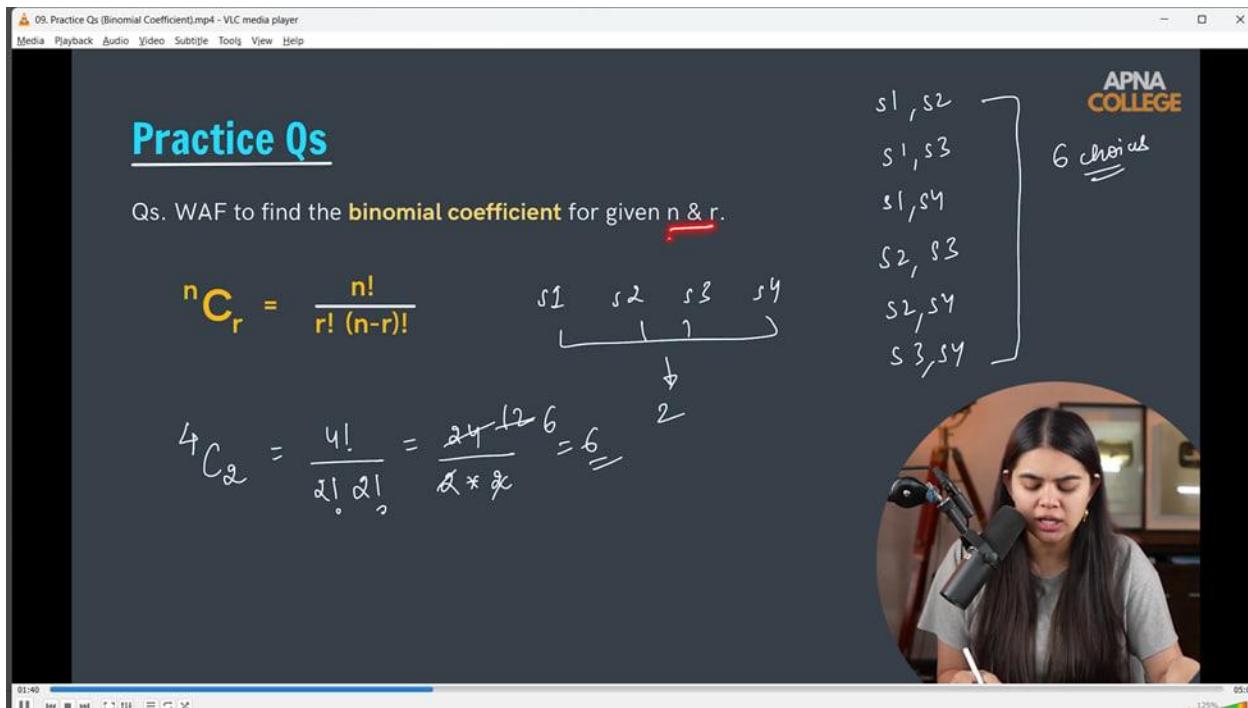
```
// Work done boss  
// */  
  
// // Forward Declaration - When a function is declared before main function and defined  
// after the main function. It's forward Declaration. Pehle hi declare kr diya and ab define kr  
// rheh  
  
// cout<<sum(2,4)<<endl; // 2,4 are arguments which are need to be pass in the parameters  
// cout<<mul(2,4)<<endl;// 8  
  
// cout<<sub(2)<<endl;//1 - when no 2nd argument passed but 2nd parameter defined then  
// by default defined 2nd parameter 1 will become the 2nd argument here.  
  
// cout<<sub(5,2)<<endl;//3 - here when, on passing both the arguments, system will not  
// consider the by default defined the 2nd parameter = 1 or any value  
  
// return 0;  
// }  
  
// _____  
  
// int Prod(int a, int b = 2)  
// {  
//     int theMultiplication = a * b;  
//     return theMultiplication;  
// }  
  
// bool isprime(int n)  
// {  
//     if (n == 1)  
//     {
```

```
//      cout << "Hey, it's exceptional dude neither prime nor composite buddy" << endl;
//  }

//  for (int i = 2; i <= sqrt(n); i++)
//  {
//      if (n % 2 == 0)
//      {
//          return false;
//      }
//      else
//      {
//          return true;
//      }
//  }
//  return isprime;
//}

// _____
// int factorial(int c)
//{
//    int fact = 1;
//    for (int i = 1; i <= c; i++)
//    {
//        fact *= i;
//    }
//    cout << "Factorial of no. - " << c << " is - " << fact << endl;
//    return fact;
```

```
// }
```



```
// int BinomialCoefficient(int n, int r)
// {
//     int val1 = factorial(n);
//     int val2 = factorial(r);
//     int val3 = factorial(n-r);

//     int result = val1 / (val2 * val3);
//     cout<<"BinomialCoeffiecent result is - "<<result<<endl;

// }

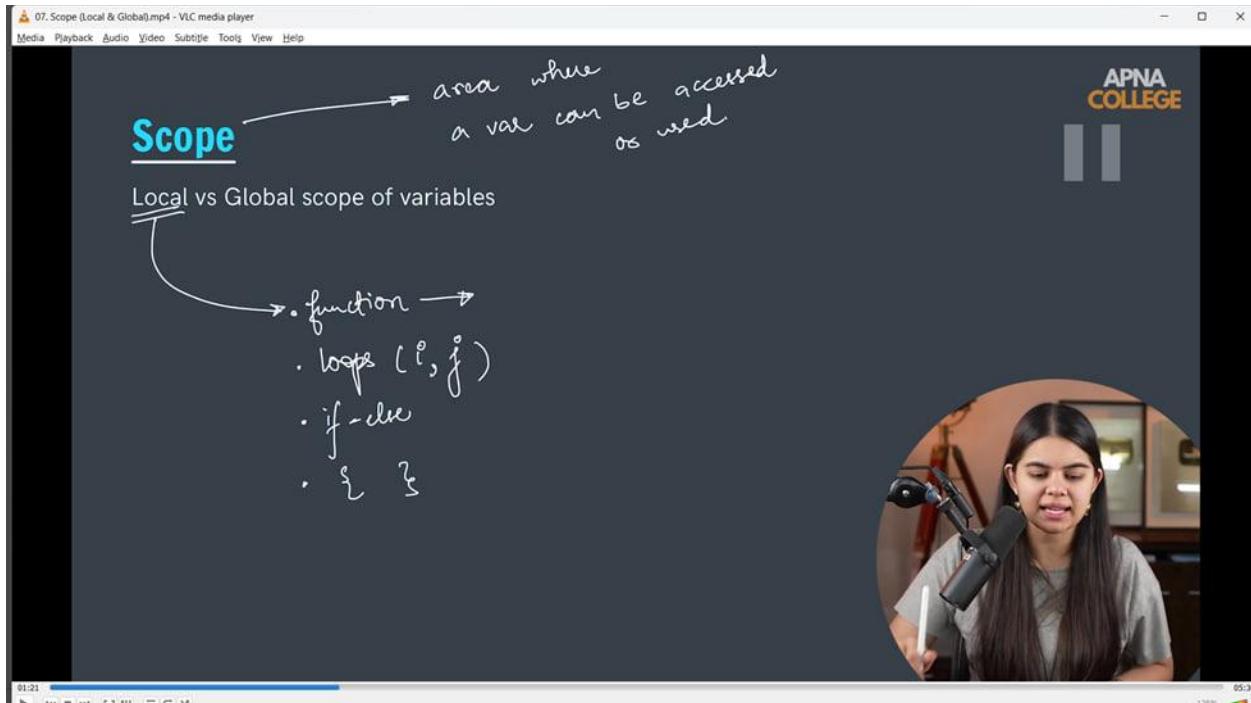
// int main()
// {
```

```

// cout << Prod(5) << endl; // 10 - by using default defined parameter 2 with 1 argument
// cout << Prod(5, 10) << endl; // 50 - by using both parameters with both arguments
// cout << isprime(23) << endl; // 1
// cout << isprime(2) << endl; // 1 - Prime Number
// cout << isprime(8) << endl; // 0 - Not a prime number
// isprime(1); // Hey, it's exceptional dude neither prime nor composite buddy
// cout << factorial(5) << endl; // 120
// factorial(10); // Factorial of no. - 10 is - 3628800
// BinomialCoefficient(4,2);
// }
// _____

```

### **/// Scoping in Function -**



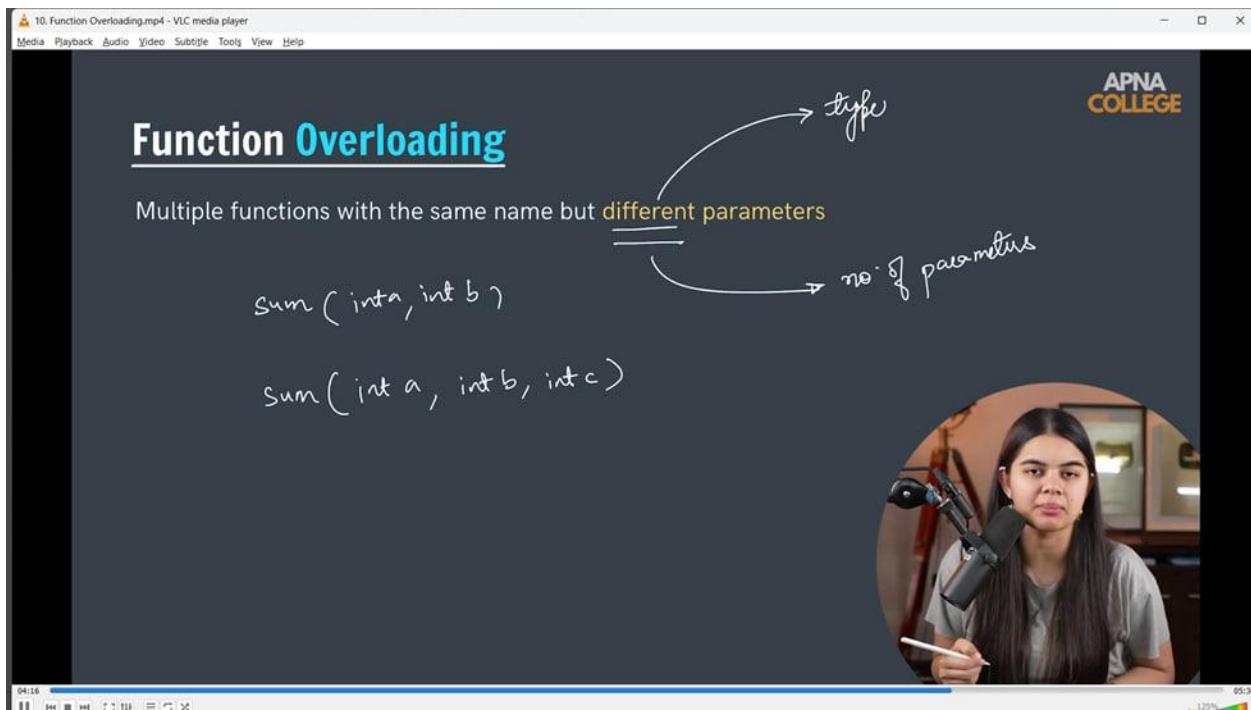
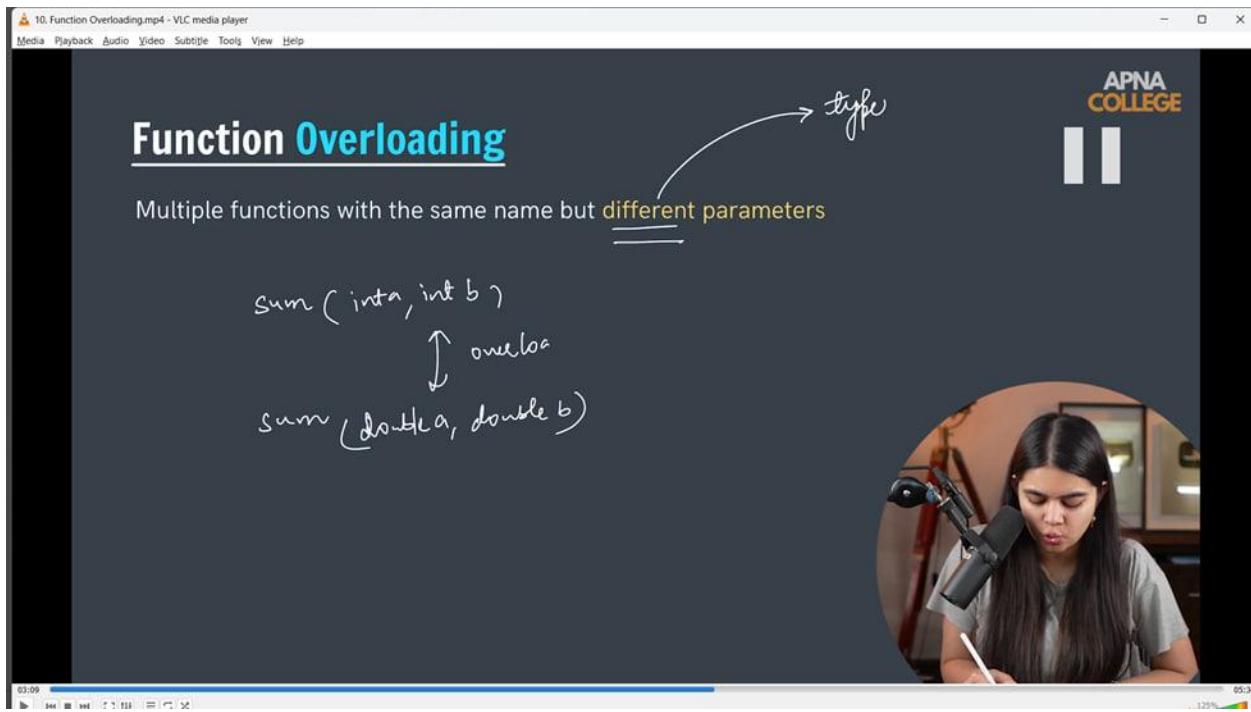
```
/*
```

// Locally Defined- If we define a variable only in the declaring function and want to use in the main function then - Not Possible as only locally declared

// Locally Defined- If we define the variable in any loop or in any if else statement then, outside of that we can't access the variable as defined within that local scope.

// Globally Defined- So, for avoiding such issues, we can define particular variable at the top just after the header files. Above the declaratory function as well as main function

```
// */  
// int num = 25;  
// int sum(int a, int b)  
// {  
//     int addition = a + b;  
//     cout << addition << endl;  
//     cout << "No. which is declared globally is - " << num << endl;  
// }  
// int main()  
// {  
//     sum(15, 10);  
//     cout << "No. which is declared globally is - " << num << endl;  
  
// /*  
// 25  
// No. which is declared globally is - 25  
// No. which is declared globally is - 25  
// */  
// _____
```



```
// int sum(int a, int b)
// {
```

```
// cout<<(a+b)<<endl;  
// return a+b;  
// }  
  
// int sum(double a, double b)  
// {  
// cout<<(a+b)<<endl;  
// return a+b;  
// }  
  
// int sum(int a, int b, int c)  
// {  
// cout<<(a+b+c)<<endl;  
// return a+b+c;  
// }  
// int main()  
// {  
// sum(45,5);//50  
// sum(12.5,2.5);//14 -  
// // it is 15 actually but due to considering as a double value compiler giving 14, so for  
overcoming trhis issuels we can use the same funciton with doffernt data gtype - it's called as  
fduncitn ovberloading  
// sum(12.5,2.5);//15 - compiler won't confuse here ki kisme value pass kare, vo datatype ke  
according kr lega, its overloading  
  
// // AND THE SAME GOES FOR multiple paramters wuith same funciton name  
// sum(5,4,3);//12
```

```
// }

// _____
// QUn - Print all prime numbers from 2 to range n

// bool isPrime(int n)

// {
//   if (n == 1)
//   {
//     cout << "No prime nor Composite" << endl;
//   }

//   for (int i = 2; i*i <=n; i++)
//   {
//     if (n % i == 0)
//     {
//       return false;
//     }
//   }

//   return true;
// }

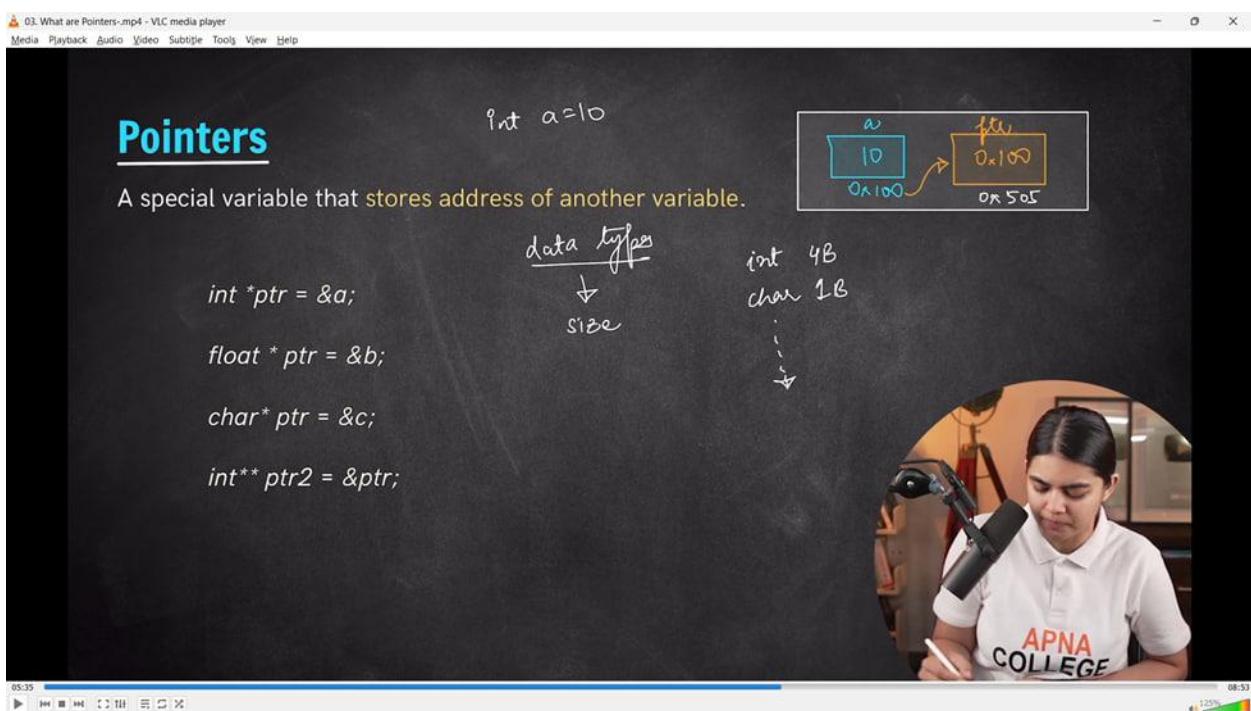
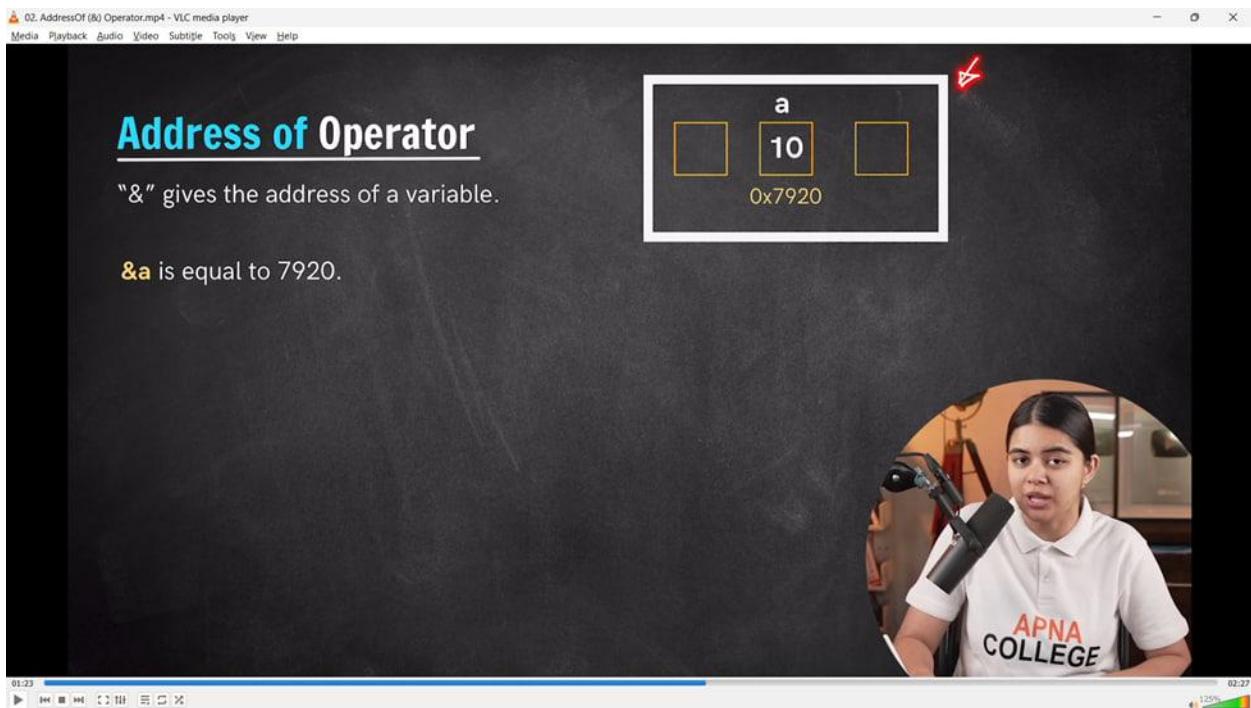
// void AllPrimes(int n)

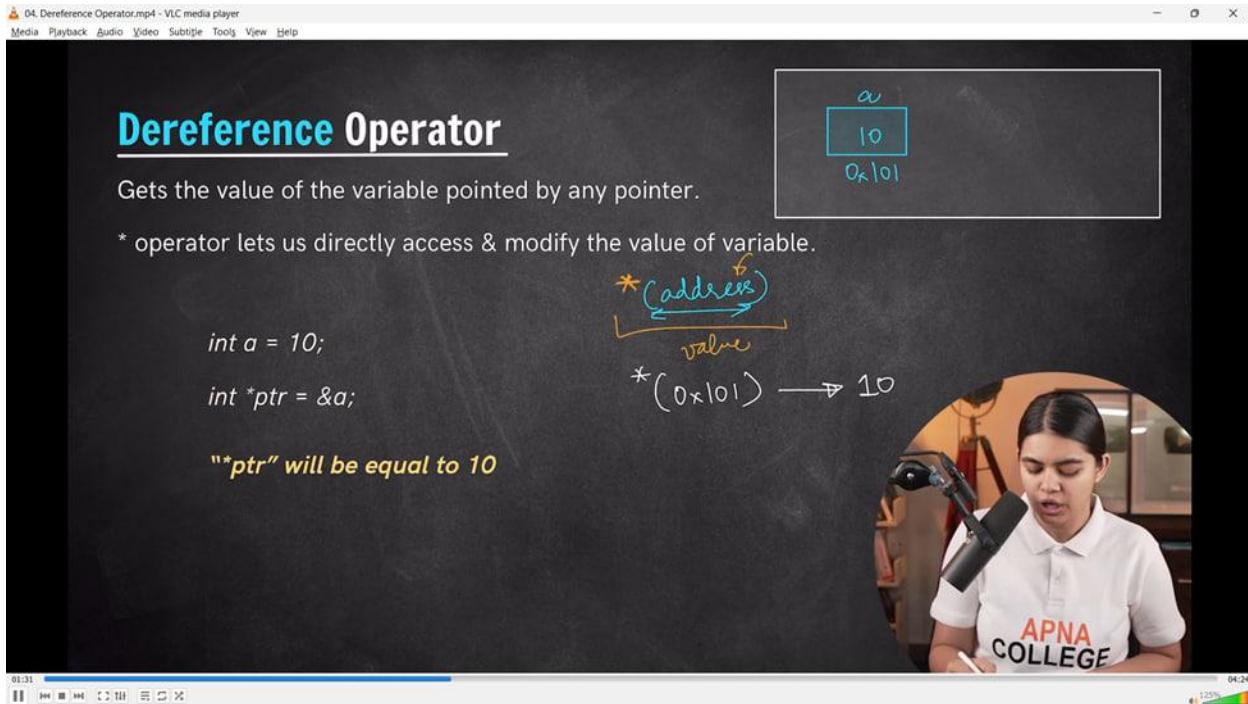
// {
//   for (int i = 2; i <= n; i++)
//   {
//     if (isPrime(i))
```

```
//      {
//          cout << i << " is prime number" << endl;
//      }
//  }
//  cout<<endl;
//}

// int main()
//{
//    AllPrimes(13);
//    return 0;
// /*
// 2 is prime number
// 3 is prime number
// 5 is prime number
// 7 is prime number
// 11 is prime number
// 13 is prime number
// */
// }

// _____
// Pointers in cpp -
```





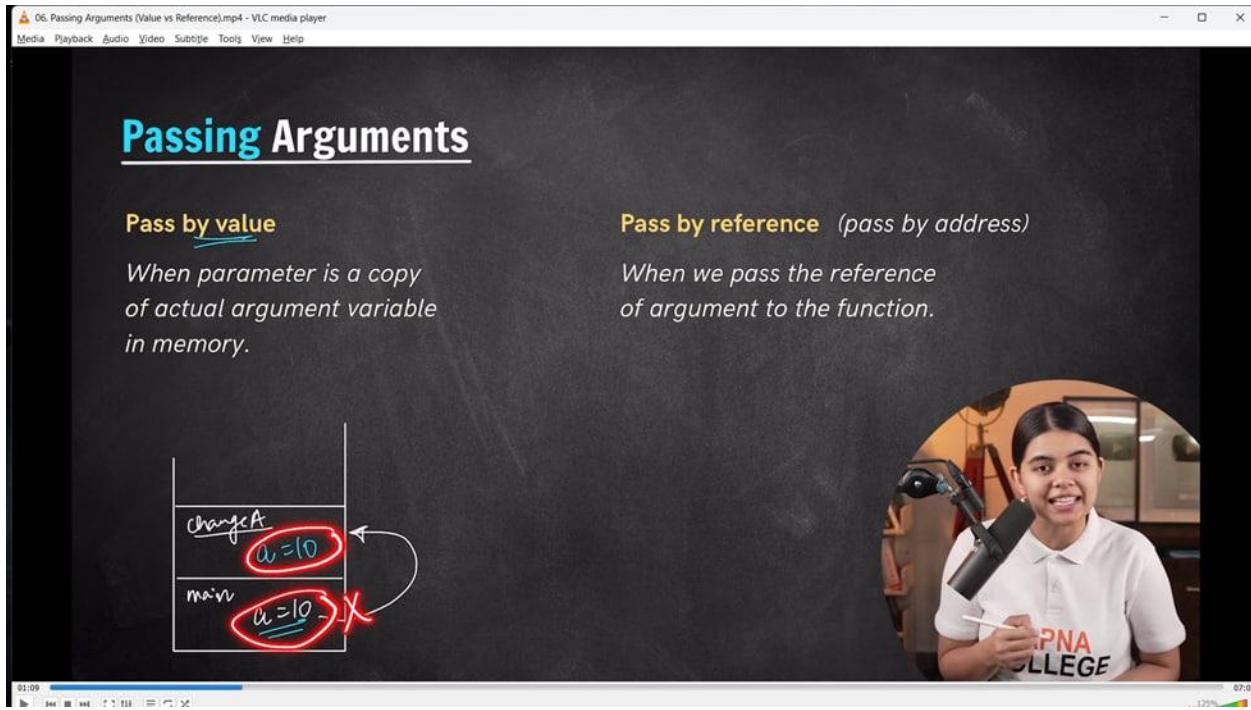
```
// int main()
//{
//    int c = 10;
//    int *ptr = &c;           // int ptr type can never store the value, it always stores the
//                            // address itself
//    cout << &c << "==" << ptr << endl; // 0x61fed4==0x61fed4

//    cout << sizeof(ptr) << endl;//4 - size of int pointer

//    float p = 3.1416;
//    float *ptr2 = &p;
//    cout << &p << "==" << ptr2 << endl;//0x61ff00==0x61ff00
//    cout << sizeof(ptr2) << endl;//4 - size of float pointer
```

```
// // Pointer of Pointer Approach -  
// int **pptr = &ptr;  
// cout<<&ptr<<"=="<<pptr<<endl;//0x61ff00==0x61ff00  
  
// // Dereference Operator -  
// int a = 20;  
// int *ptr3 = &a;  
// cout<<&a<<endl;//0x61fef8  
// cout<<*(&a)<<endl;//20  
// cout<<*ptr3<<endl;//20  
  
// // Null Pointer  
// // if pointer is not pointing the address of any variable then this will give Garbage Value  
  
// int *ptr4;  
// cout<<ptr4<<"\n";//0x76e18c1d - Automatically gives Garbage Value  
  
// int *ptr5=NULL;  
// cout<<ptr5<<endl;//0x74e18c1d  
// cout<<*ptr5<<endl;//0  
// }
```

## Passing Arguments



### Pass by Value –

```
// Passing Arguments in funciton -
```

```
// Pass by value - when parameter is a copy of actual arguement variable in memory
```

```
// void changeA(int a)  
// {  
//     a=30;  
//     cout<<a<<endl;  
// }
```

```
// int main()  
// {
```

```
// int a = 10;  
  
// changeA(a);//30 - Funciton call kiya he to function declaratin me jaa kr function me  
variable ki assigned value ko call krega  
  
// cout<<a<<endl;//10 - vapas se main fun me aa gya h to iss scope me jo variable defined h  
usko value ko print krega  
  
// }
```

// Pass by Reference - using pointer

```
// void changeB(int *ptr)  
  
// {  
//     *ptr = 20;  
//     cout << *ptr << endl;  
// }  
  
// int main()  
  
// {  
//     int a = 10;  
//     changeB(&a);//20 - call by address he. Ptr me &a h and function declaration me us pointer  
se uski valuie assign hui he. to calling me funciton ki under variable ki assigned value print hogi  
  
//     cout << a << endl;//20 - return to main function - to uske ander var ki jo value defined he  
vo print hogi  
// }
```

## // Pass by Reference - using Reference Variables

The screenshot shows a VLC media player window with a slide titled "Reference Variables". The slide contains the following text and diagram:

```
int a = 5;
int &b = a;
```

*a & b refer to the same location in memory*

A diagram in the top right corner shows a box containing the number "5" with two arrows pointing to it from labels "a" and "b".

On the right side of the slide, there is a circular inset image of a young woman with dark hair, wearing a white shirt, speaking into a microphone. The background behind her shows an indoor setting with a lamp and some furniture.

```
// void changeC(int *ptr)
//{
//    *ptr = 20;
//    cout<<*ptr<<endl;
//}
```

```
// void changeD(int &c)
//{
//    c=20;
//    cout<<c<<endl;
//}
// int main()
//{
//}
```

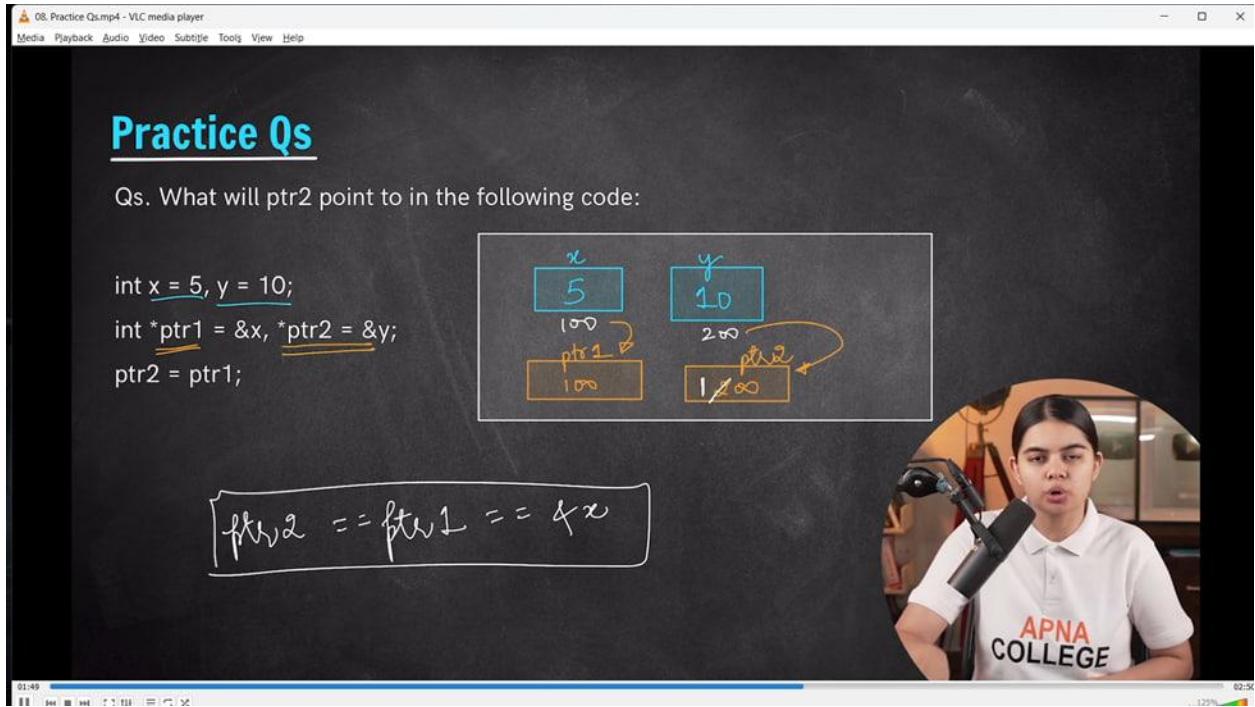
```

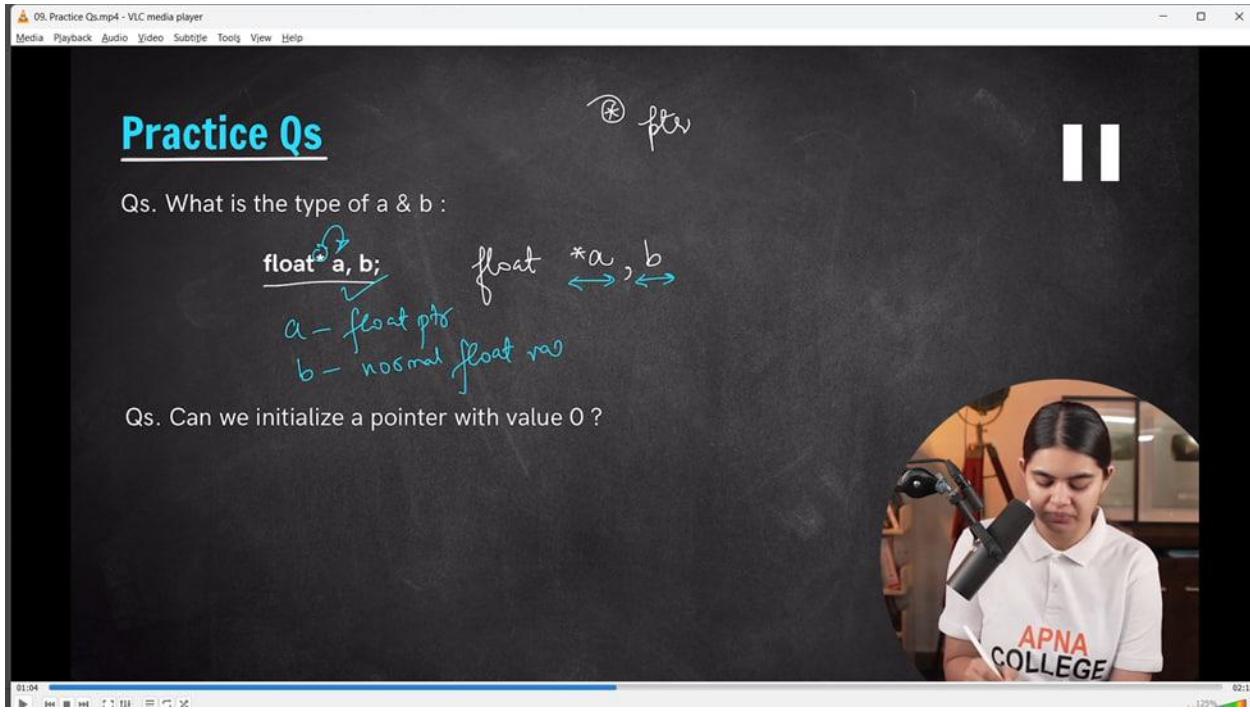
// int a = 10;
// int &b = a;
// b=25;
// cout<<b<<endl;//25
// cout<<a<<endl;//25

// int c =10;
// changeD(c);//20
// cout<<c<<endl;//20 - yaha pr pointer ke saath reference diya h fiun declaration me to main
value me bhi change aayag
// // This is call by reference using reference variable. memory will back to main function but
as we're used reference var, so there will also change in main fun too.

// }

```





// Qun -

```
// int main()
//{
//    int x=5, y=10;
//    int *ptr1 = &x, *ptr2 = &y;
//    ptr2=ptr1;

//    cout<<ptr2<<endl;//0x61ff04
//    cout<<ptr1<<endl;//0x61ff04
//    cout<<&x<<endl;//0x61ff04

//    // What os the typr of the following variables -
//    float*a,b;
//    // a - float pointer, b - normal float variable
```

// }

## Number System in Programming–

### Binary to Decimal –

02. Convert Binary to Decimal.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

### Binary Number System

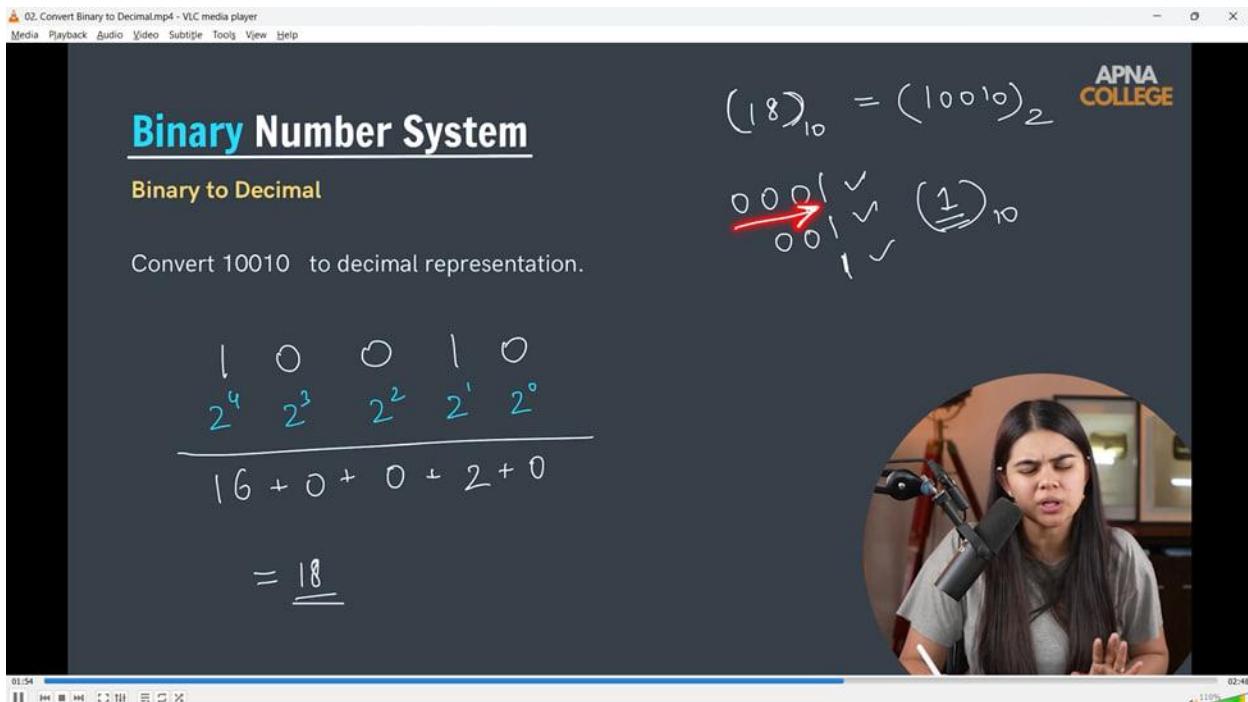
#### Binary to Decimal

Convert 10010 to decimal representation.

$$\begin{array}{r} 1 \quad 0 \quad 0 \quad 1 \quad 0 \\ 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \hline 16 + 0 + 0 + 2 + 0 \\ = 18 \end{array}$$

$(18)_{10} = (10010)_2$

APNA COLLEGE



03. Convert Decimal to Binary.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

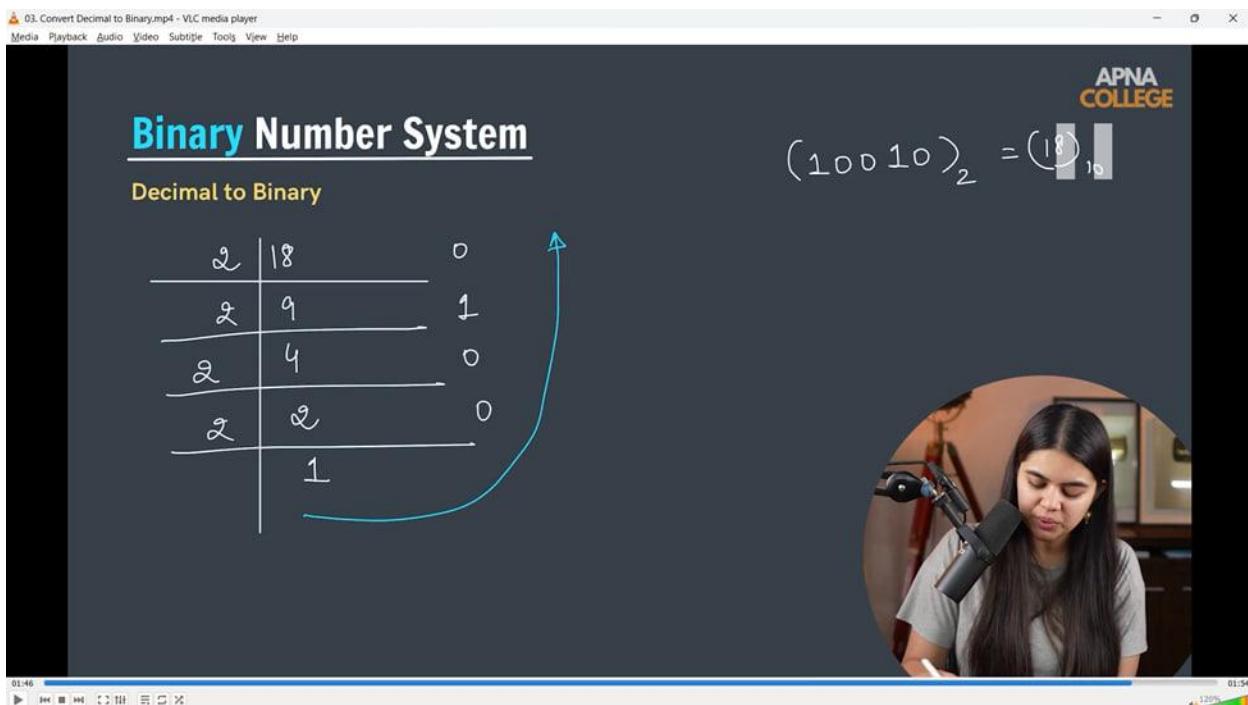
### Binary Number System

#### Decimal to Binary

$$\begin{array}{r} 18 \\ 2 | 18 \quad 0 \\ 2 | 9 \quad 1 \\ 2 | 4 \quad 0 \\ 2 | 2 \quad 0 \\ 1 \end{array}$$

$(10010)_2 = (18)_{10}$

APNA COLLEGE



04. Question (Decimal to Binary).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Binary Number System

**Decimal to Binary**

Convert 37 to binary representation.

$(100\ 101)_2 = (37)_{10}$

APNA COLLEGE

00:49 11:33 125%

04. Question (Decimal to Binary).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Binary Number System

**Decimal to Binary**

Convert 37 to binary representation.

$$\begin{array}{r} 2^6 \\ \hline 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ \hline 1 & 0 & 0 & 1 & 0 & 1 \end{array} \quad (32) + 4 + 1 = \underline{\underline{37}}$$

$(100\ 101)_2 = (37)_{10}$

APNA COLLEGE

03:40 11:33 125%

04. Question (Decimal to Binary).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

1        1  
 2        1 0  
 3        1 1  
 4        1 0 0  
 5        1 0 1  
 6        1 1 0  
 7        1 1 1  
 8        1 0 0 0  
 9        1 0 0 1  
 10      1 0 1 0  
 11      1 0 1 1  
 12      1 1 0 0  
 13      1 1 0 1  
 14      1 1 1 0  
 15      1 1 1 1  
 16      1 0 0 0 0

Decimal to Binary (0 to 16) APNA COLLEGE

range → size to data  
~~int x = 3;~~

$\downarrow$

$16(5)$

11:05 11:33 125%

## Data Type Modifiers & Range of Int –

06. Data Type Modifiers.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Data Type Modifiers

Alter the meaning of existing data types.

**long**     $\geq 4$  bytes (more than int)

**short**    2 bytes

**signed**    signed int is same as int

**unsigned**    can only store non-negative numbers

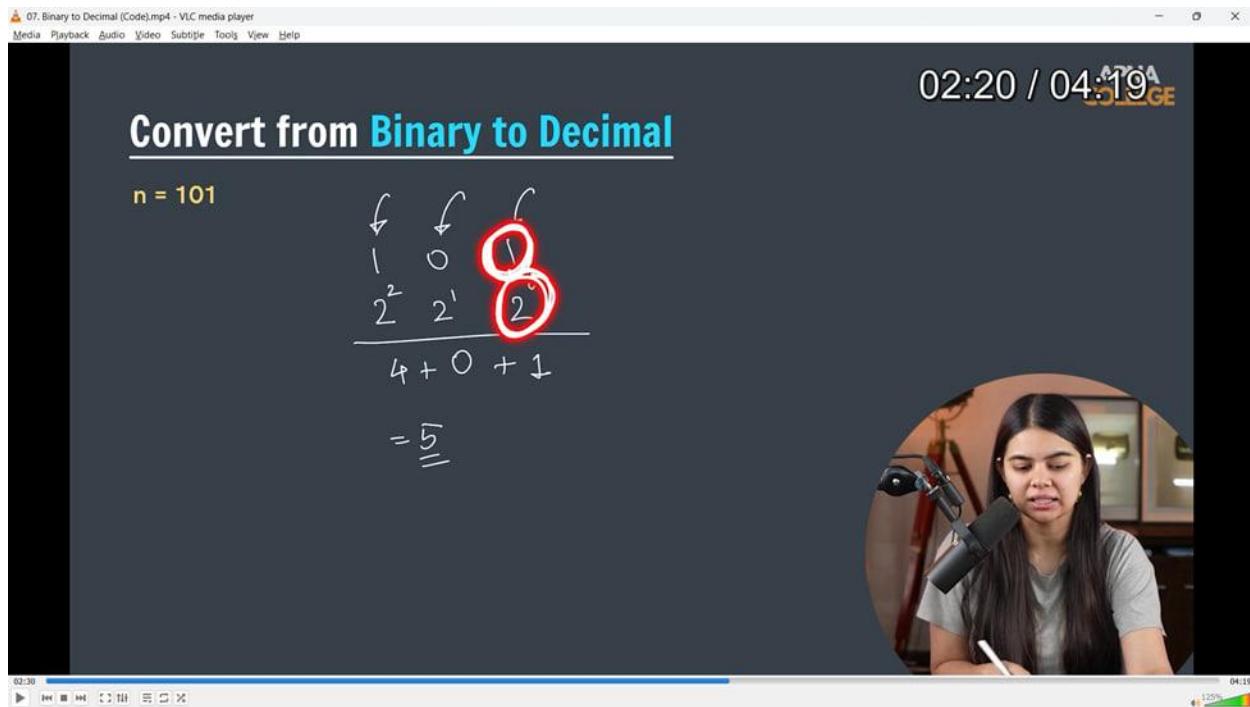
**long long**    same as long long int

$\text{int} \rightarrow -2^{31} \text{ to } 2^{31}-1$   
 $(4 \text{ bytes})$   
 $\underline{\underline{32 \text{ bits}}}$

APNA COLLEGE

02:59 11:12 125%

## Binary to Decimal



```
// void binToDec(int binNum)
//{
//    int n = binNum;
//    int decNum = 0;
//    int pow = 1; // 2^0 2^1 2^2 2^3..

//    while (n > 0)
//    {
//        int lastDig = n % 10;
//        decNum += lastDig * pow;
//        pow = pow * 2;
//        n = n / 10;
//    }
}
```

```
// cout << decNum << endl;  
// }  
// int main()  
// {  
// // Size of for different int data types -  
// // cout<<sizeof(int)<<endl;//4  
// // cout<<sizeof(long int)<<endl;//4  
// // cout<<sizeof(short int)<<endl;//2  
// // cout<<sizeof(long long int)<<endl;//8  
  
// binToDec(101);//5  
// binToDec(1011);//11  
// binToDec(1111);//15  
// }
```

---

## Decimal to Binary -

The video player interface shows the following details:

- Title: Convert from Decimal to Binary
- Video duration: 03:26
- Player controls: Back, Forward, Stop, Play, Volume, etc.
- APNA COLLEGE logo in the top right corner.
- Handwritten notes on the right side of the screen:
  - ① rem / 2
  - ② rem \* 10<sup>0</sup> ⇒ 0
  - ③ pow = pow \* 10
  - ④ m / 2
  - 4 / 2 ⇒ 0 \* 10<sup>0</sup>
  - 2 / 2 ⇒ 0 \* 10<sup>0</sup>
  - 1 / 2 ⇒ 1 \* 10<sup>0</sup>
  - 0
- Below the notes:
$$\frac{100}{\downarrow} = 1 \cdot 10^2 + 0 \cdot 10^1 + 0 \cdot 10^0$$
- Code snippet on the left:

```
void DecToBin(int DecNum)
{
    int n = DecNum;
    int pow = 1; // 10^0, 10^1, 10^2...
    int BinNum = 0;

    while (n > 0)
    {
        int rem = n % 2;
```

```
void DecToBin(int DecNum)
```

```
{
```

```
    int n = DecNum;
```

```
    int pow = 1; // 10^0, 10^1, 10^2...
```

```
    int BinNum = 0;
```

```
    while (n > 0)
```

```
{
```

```
        int rem = n % 2;
```

```
    BinNum += rem * pow;  
    n = n / 2;  
    pow = pow * 10;  
}  
  
cout<<BinNum<<endl;  
}
```

```
int main()  
{  
    DecToBin(15);//1111  
    DecToBin(4);//100  
}
```

/\*

Do the comparision b/w both the methods -

in Bin to Dec - Modulus by 10 as well as devide by 10 and pow of 2, while

in Dec to Bin - Modulues/Remainder by 2 as well as divide by 2 and pow of 10

\*/

// \_\_\_\_\_

## **Arrays-**

- 1) Initiation Methods of an array and Output & input of an array -
- 2) Largest Value/Max Element in the array & 2.1) Min Element -
- 3) Dereferences of Pointer concept in Array -
- 4) Linear Search in array -
- 5) Reverse an array -
  - 5.1) using extra space
- 5.2 - W/o using Extra Space1 - 2 POINTERS APPROACH
- 6) Binary Search - It always for the Sorted Array
- 7) Array Pointer -
  - 7.1. Pointer Arithmetic Approaches -
  - 7.2 - Addition & Subtraction of Constants -
  - 7.3- Addition & Subtraction of Pointers
  - 7.4 - Comparison Operators
- 8) Subarrays -
- 9) Max Subarray Sum -
  - 9.1 - Clearly Brute Force Approach -
  - 9.2 - Slightly Optimized approach - for decreasing the time complexity
  - 9.3 - Using Most Optimized Approach - Kadane's Algorithms. For very less Time Complexity
- 10) buy & sell stock problem-
- 11) Trapping Rainwater Problem -

## 2) Array -

**Arrays in cpp** - Linear Collection of Same Type of Elements that are stored together in Contiguous Memory Spaces.

### //1) Initialization Methods of an array -

The screenshot shows a video player window for '01. Basics of Array.mp4' in VLC media player. The video content is a chalkboard lesson. At the top left, it says 'Arrays'. A handwritten note above the board says 'Index → position'. The board defines an array as a 'Linear collection of **same type** of elements that are stored together in **contiguous memory** spaces.' Below this, there is a diagram of an array named 'marks' containing integer values: 99, 78, 62, 54, 88, 94, 95. The indices 0 through 6 are shown below the array. A handwritten note above the array says 'int array'. To the right of the array, there is a handwritten note: 'length / size ⇒ 7' with an arrow pointing to the value 7, and '0 to 6' with an arrow pointing to the index 0. On the right side of the chalkboard, there is a circular video feed of a person speaking into a microphone. The VLC player interface at the bottom shows the progress bar and other controls.

The screenshot shows a video player window for '02. Creating an Array.mp4' in VLC media player. The video content is a chalkboard lesson. At the top left, it says 'Creating an Array'. A handwritten note above the board says 'int marks[50]; → **garbage**'. Below this, there are three lines of code: 'int marks[50] = {1, 2, 3};' and 'int marks[] = {1, 2, 3};'. A handwritten note below the board says 'Memory is **statically allocated** (at compile time)'. To the right of the board, there is a diagram of an array with indices 0, 1, 2, 3, ..., 49. Brackets above the array indicate its size. Below the array, there is a diagram of an array with indices 0, 1, 2, 3, ..., 49, where the first four slots contain values (1, 2, 3, 0) and the rest are empty, labeled 'garbage value'. On the right side of the chalkboard, there is a circular video feed of a person speaking into a microphone. The VLC player interface at the bottom shows the progress bar and other controls.

```

// int marks[50];           // 0 to 49

// cout << marks[0] << endl; // 4199136 - whn we don;t innitilize the array with any value it
takes any garbage value in the memory

// cout << marks[1] << endl; // 17566816

// cout << marks[2] << endl; // 17566816

// cout << marks[50] << endl; // 4201056

// int percentage[5] = {0, 1, 2, 3, 4};

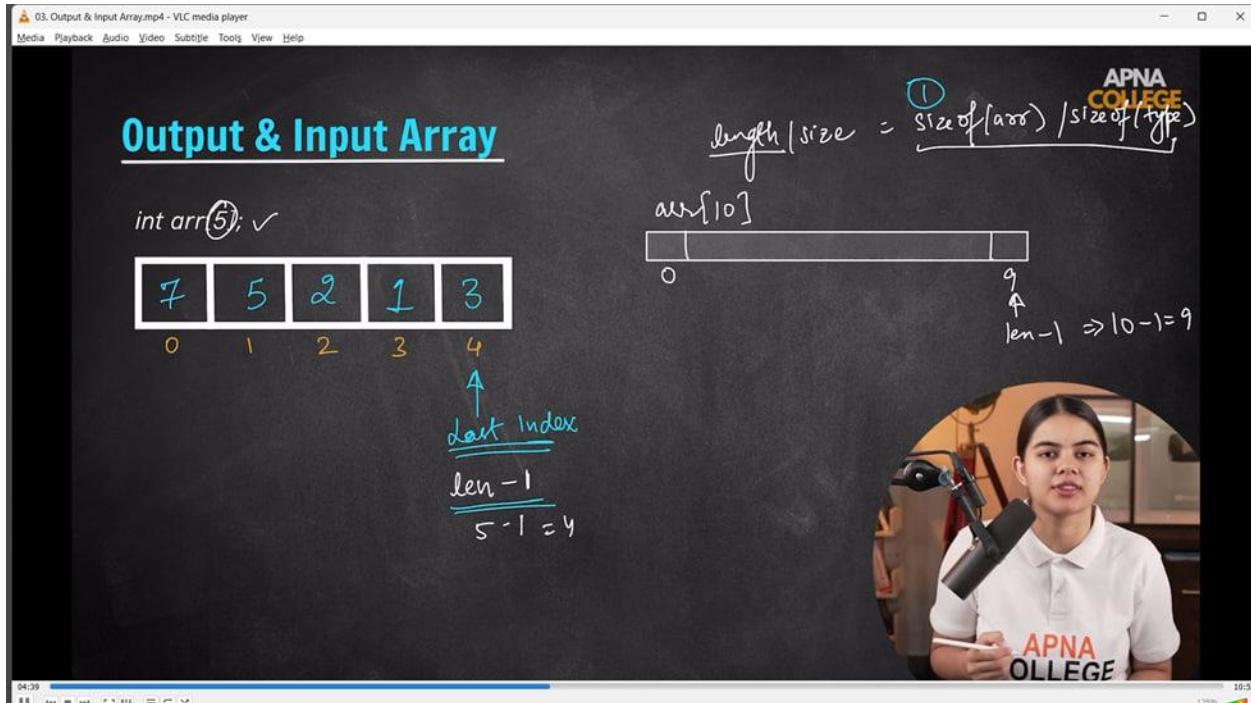
// cout << percentage << endl;

// cout << sizeof(percentage) << endl;                                // 20 - int size = 4 bytes , so 5 int =
4*5 = 20 bytes

// cout << "length of array is - " << sizeof(percentage) / sizeof(int) << endl; // length of array is -
5

```

### // 1.1)// Ouput & input of an array -



### // 1.1)// Ouput & input of an array -

```
// int arr[5] = {0, 1, 2, 3, 4};

// for (int i = 0; i < 5; i++)
//{
//    cout << i << " "; // 0 1 2 3 4
//}

// cout << endl;

// int len = sizeof(arr) / sizeof(int);

// for (int idx = 0; idx <= len - 1; idx++)
//{
//    cout << arr[idx] << " "; // 0 1 2 3
//}

// cout << endl;
// _____
// 2) Largets Value/Max Element in the array -

// int n;
// cout << "n's value - " << endl;
// cin >> n;

// cout << "ENte array values - " << endl;
// int arr2[n];

// for (int i = 0; i < n; i++)
//{

```

```
//    cin >> arr2[i];  
// }  
  
// cout << "Entered array's value is - " << endl;  
// for (int i = 0; i < n; i++)  
// {  
//    cout << arr2[i] << " ";  
// }  
// cout << endl;  
  
// int maxEle = arr2[0];  
  
// for (int i = 0; i < n; i++)  
// {  
//    if (arr2[i] > maxEle)  
//    {  
//        maxEle = arr2[i];  
//        cout << "Current assigned max value is - " << arr2[i] << endl;  
//    }  
// }  
// cout << "So, Finally max value is - " << maxEle << endl;  
// _____  
// 2.1) Min Element -  
//    int n;  
//    cout << "n's value - " << endl;  
//    cin >> n;
```

```
// cout << "ENte array values - " << endl;  
// int arr2[n];  
  
// for (int i = 0; i < n; i++)  
// {  
//     cin >> arr2[i];  
// }  
  
// cout << "Entered array's value is - " << endl;  
// for (int i = 0; i < n; i++)  
// {  
//     cout << arr2[i] << " ";  
// }  
// cout << endl;  
  
// // For Min Element  
// int minEle = arr2[0];  
  
// for (int i = 0; i < n; i++)  
// {  
//     if (arr2[i] < minEle)  
//     {  
//         minEle = arr2[i];  
//         cout << "Current assigned min value is - " << arr2[i] << endl;  
//     }  
// }
```

```
// }

// cout << "So, Finally min value is - " << minEle << endl;

// /*
// n's value -
// 6

// ENte array values -
// 89 45 23 12 56 78

// Entered array's value is -
// 89 45 23 12 56 78

// Current assigned min value is - 45

// Current assigned min value is - 23

// Current assigned min value is - 12

// So, Finally min value is - 12

// */

// }
```

**// 3) Dereferences of Pointer concept in Array -**

05. Arrays are passed by Reference.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Arrays are passed by reference

```
void printArr( int arr[ ] ) { ... } } same  
void printArr( int *arr ) { ... }
```

C++ array name → pointer

APNA COLLEGE

04:04 12:47

05. Arrays are passed by Reference.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Arrays are passed by reference

```
void printArr( int arr[ ] ) { ... } } same  
void printArr( int *arr ) { ... }
```

C++ array name → pointer

APNA COLLEGE

09:27 12:47

```
// Array's are by default passed by reference. The value change in function affects in main fun also
```

```
// void func(int arr[])
//{
//    arr[0] = 1000;
//    cout << arr[0] << endl;
//}

// void func2(int *ptr)
//{
//    ptr[0] = 1000;
//    cout << ptr[0] << endl;
//}

// void PrintArr(int arr2[])
//{
//    int n = sizeof(arr2)/sizeof(int);
//    for (int i = 0; i < n; i++)
//    {
//        cout << arr2[i] << " ";
//    }
//    cout << endl;
//}

// void PrintArr2(int arr2[], int n)
//{

```

```

// // int n = sizeof(arr2)/sizeof(int);

// for (int i = 0; i < n; i++)

// {

//     cout << arr2[i] << " ";

// }

// cout << endl;

// }

// int main()

// {

//     int a = 5;

//     int *ptr = &a;

//     cout << ptr << endl; // 0x61ff08

//     cout << *ptr << endl; // 5 - Derefernce operation - using pointer getting variables value

//     int arr[] = {2, 4, 6, 8, 10};

//     cout << *arr << endl; // 2 - can also got by usng cout<<arr[0]. Here using Aestic as its
//     dererfenrece of array

//     cout << *(arr + 1) << endl; // 4

//     cout << *(arr + 2) << endl; // 6

//     func(arr); // 1000 - jb fun decl me array pass kiya jaata he to yh by default CALL BY
//     REFERENCE use krta h

//     cout << arr[0] << endl; // 1000 - Passing array name is eq. to passing the pointer

//     func2(ptr); // 1000

//     // so clearly, pointer and array are quivalent. in case of array

```

```
// int arr2[] = {5, 10, 15, 20};

// PrintArr(arr2); // w/o paasing the array size too in paramter we can't print thte array.
Because it consider as thee pointer size not the array size

// /*
// 1_IntroToArray.cpp: In function 'void PrintArr(int*)':

// 1_IntroToArray.cpp:150:24: warning: 'sizeof' on array function parameter 'arr2' will return size
of 'int*' [-Wsizeof-array-argument]

// int n = sizeof(arr2)/sizeof(int);

//           ^
// 1_IntroToArray.cpp:147:24: note: declared here

// void PrintArr(int arr2[])

// */

// PrintArr2(arr2,4); // 5 10 15 20 - so make sure array apne aap me ek pointer h which by
defualt use sthe Call by reference propeerty. and funciton me array pass krte tym, hmesha size
bhi as a paramter pass kre

// }

// _____
```

**//4) Linear Search in array -**

07. Linear Search.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Linear Search

search for key = 10

arr

2	4	6	8	10	12	14	16
↑ 0	↑ 1	↑ 2	↑ 3	↑ 4			

(arr, n, key) {  
for (i=0 to n) {  
if (arr[i] == key) {  
return i;  
}  
}

APNA COLLEGE

07. Linear Search.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Linear Search

search for key = 10

2	4	6	8	10	12	14	16
↑ 0	↑ 1	↑ 2	↑ 3	↑ 4			

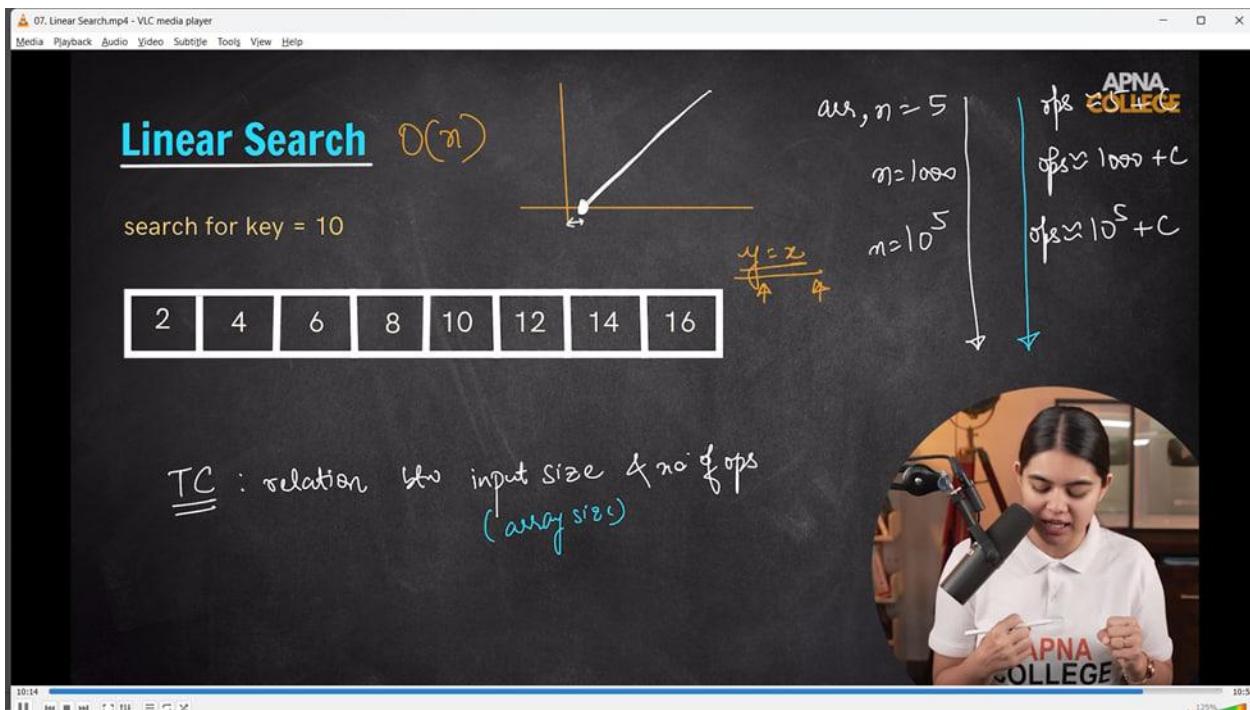
$y = x + C$

$arr, n = 5$   
 $m = 10^0$   
 $m = 10^5$

$ops \approx 10^0 + C$   
 $ops \approx 10^5 + C$

$\underline{TC}$  : relation b/w input size & no. of ops  
(array size)

APNA COLLEGE



// my method -

```

// int linearSearch()

//{
//    int n;
//    int key;
//    cout << "No of elements in Array - " << endl;
//    cin >> n;

//    int arr[n];
//    cout << "What are the array's elements" << endl;
//    for (int i = 0; i < n; i++)
//    {
//        cin >> arr[i];
//    }

```

```
// cout << "So, the entered array is -" << endl;
// for (int i = 0; i < n; i++)
// {
//     cout << arr[i] << " ";
// }
// cout << endl;

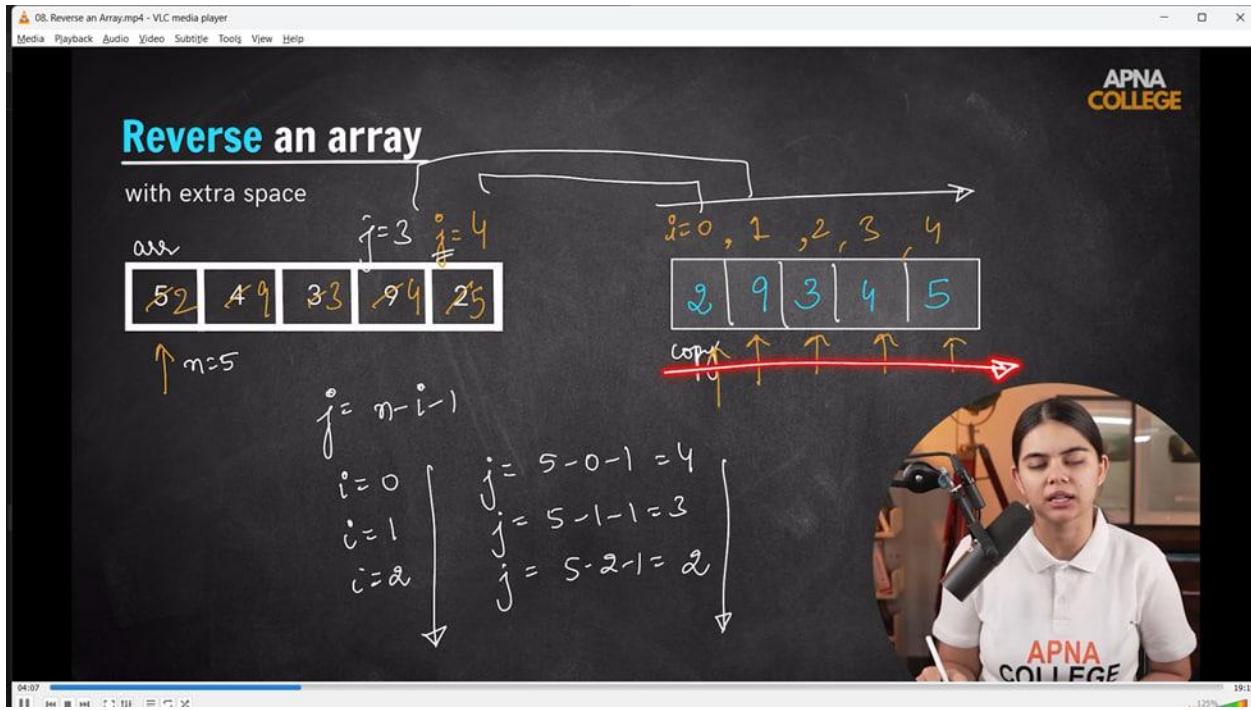
// cout << "Now mention the element you want to search for -" << endl;
// cin >> key;

// for (int i = 0; i < n; i++)
// {
//     if (arr[i] == key)
//     {
//         cout << "So, key is - " << key << " and found at " << i << endl;
//     }
// }
// }

// int main()
// {
//     linearSearch();
// }
// _____
```

```
// using proper method -  
  
// int linearSearch(int arr[], int key)  
// {  
//     for (int i = 0; i < 8; i++)  
//     {  
//         if (arr[i] == key)  
//         {  
//             return i;  
//         }  
//     }  
//     return -1;  
// }  
  
// int main()  
// {  
//     int arr[8] = {8, 16, 24, 32, 40, 48, 56, 64};  
//     cout<<linearSearch(arr,15)<<endl;//-1  
//     cout<<linearSearch(arr, 40)<<endl;//4  
//     return 0;  
  
// // T.C -is simply depends on no. of iterations which is depend on no. of elements - O(n)  
// }  
// _____
```

## //5) Reverse an array



### // 5.1) using extra space

```
// int main()
//{
//    int n;
//    cout << "value of array size" << endl;
//    cin >> n;

//    int arr[n];
//    cout << "Write down the array elements - " << endl;
//    for (int i = 0; i < n; i++)
//    {
//        cin >> arr[i];
//    }
```

```
// cout << "So, the inserted array is - " << endl;
// for (int i = 0; i < n; i++)
// {
//     cout << arr[i] << " ";
// }
// cout << endl;

// int copyArr[n]; // an empty new array for getting all previous arra's value
// for (int i = 0; i < n; i++)
// {
//     int j = n - i - 1;
//     copyArr[i] = arr[j];
// }

// // now elements are stored in copyarr in reverse ordedr, so storing in the origina again

// for (int i = 0; i < n; i++)
// {
//     arr[i] = copyArr[i];
// }

// for (int i = 0; i < n; i++)
// {
//     cout << arr[i] << " ";
// }
// cout << endl;
```

```
//    return 0;  
// /*  
// value of array size  
// 5  
// Write down the array elemenets -  
// 56 89 12 23 78  
// So, the inserted array is -  
// 56 89 12 23 78  
// 78 23 12 89 56  
// */  
// }
```

// Same approach using functions -

```
// void reverseArray(int arr[], int n)  
// {  
  
//     for (int i = 0; i < n; i++)  
//     {  
//         cout << arr[i] << " ";  
//     }  
//     cout << endl;  
// }  
  
// int main()
```

```
// {  
//     int n;  
//     cout << "value of array size" << endl;  
//     cin >> n;  
  
//     int arr[n];  
//     cout << "Write down the array elemenets - " << endl;  
//     for (int i = 0; i < n; i++)  
//     {  
//         cin >> arr[i];  
//     }  
//     cout << "So, the inserted array is - " << endl;  
//     for (int i = 0; i < n; i++)  
//     {  
//         cout << arr[i] << " ";  
//     }  
//     cout << endl;  
  
//     int copyArr[n]; // an empty new array for getting all previous arra's value  
//     for (int i = 0; i < n; i++)  
//     {  
//         int j = n - i - 1;  
//         copyArr[i] = arr[j];  
//     }  
  
//     // now elements are stored in copyarr in reverse ordedr, so storing in the origina again
```

```

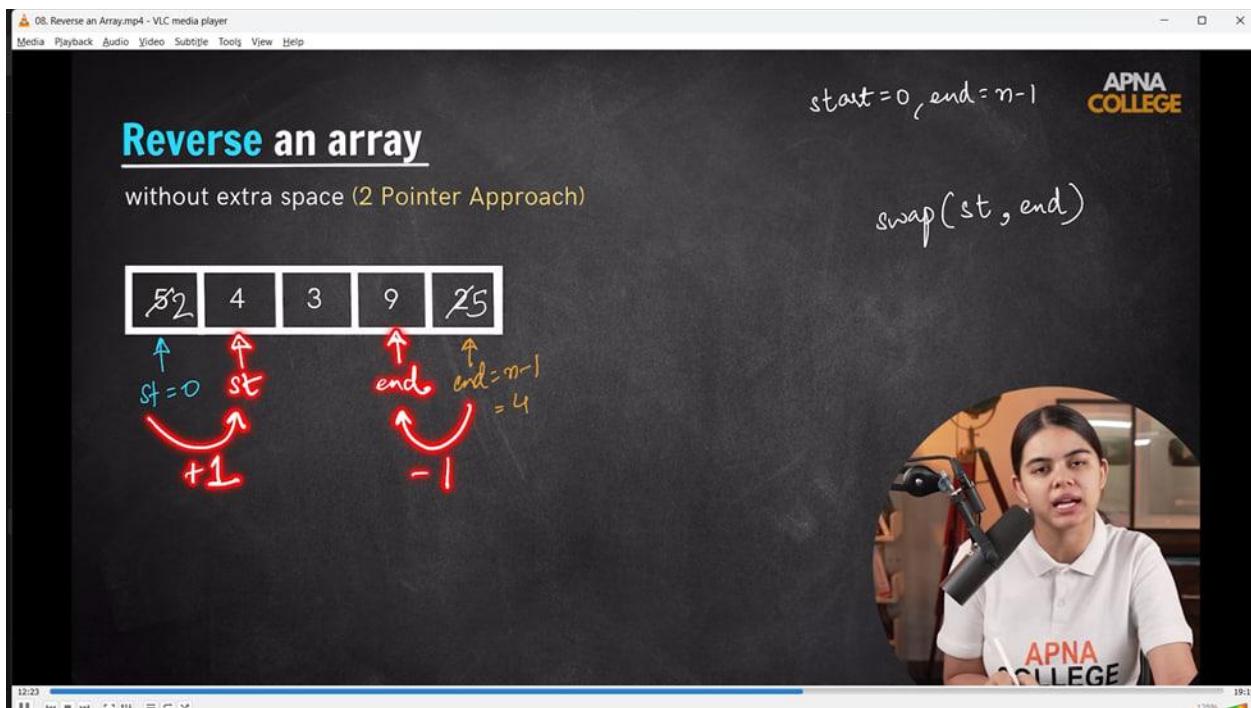
//   for (int i = 0; i < n; i++)
//   {
//     arr[i] = copyArr[i];
//   }

// reverseArray(arr, n);
// /*
// ✎ - T. C - is same - O(n) - depends on iterations which depend on no. of elements
// S.C - O(n) for Original array, but due to creation another array which is extra space also O(n) -
// so S.C - O(n*n) = O(n^2)

// */
// _____

```

## // 5.2 - W/o using Extra Space1 - 2 POINTERS APPROACH



08. Reverse an Array.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Reverse an array

SC: O(1)  
TC: O(n)

without extra space (2 Pointer Approach)

```
start = 0, end = n-1
while(start < end) {
    swap(st, end);
    start++;
    end--;
}
```

15:43 100% 125% 19:15

08. Reverse an Array.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Reverse an array

SC: O(1)  
TC: O(n)

without extra space (2 Pointer Approach)

```
int temp = a;
a = b;
b = temp;
```

17:34 100% 125% 19:15

```
// void printArray(int arr[], int n)
//{
//    cout << "Hence, the reversed array is - " << endl;
```

```
// for (int i = 0; i < n; i++)
// {
//     cout << arr[i] << " ";
// }
// cout << endl;
// }

// int main()
// {
//     int n;
//     cout << "value of array size" << endl;
//     cin >> n;

//     int arr[n];
//     cout << "Write down the array elemenets - " << endl;
//     for (int i = 0; i < n; i++)
//     {
//         cin >> arr[i];
//     }
//     cout << "So, the inserted array is - " << endl;
//     for (int i = 0; i < n; i++)
//     {
//         cout << arr[i] << " ";
//     }
//     cout << endl;

// // usingn 2 pointer approch -
```

```
// int start = 0, end = n - 1;  
// while (start < end)  
// {  
  
//     swap(arr[start], arr[end]);  
  
//     /* or by usign swap method  
//     int temp = arr[start];  
//     arr[start] = arr[end];  
//     arr[end] = temp;  
//     */  
  
//     start++;  
//     end--;  
// }  
  
// printArray(arr, n);  
// return 0;  
// /*  
// value of array size  
// 5  
// Write down the array elemenets -  
// 89 56 45 23 12  
// So, the inserted array is -  
// 89 56 45 23 12  
// Hence, the reversed array is -
```

```
// 12 23 45 56 89
// */
// }
```

## //6) Binary Search - It always works for the Sorted Array

09. Binary Search.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Binary Search** *better w/ TC*

prerequisite : sorted array

key = 12

2	4	6	8	10	12	14	16
0	1	2	3	4	5	6	7

mid

①  $\underline{mid} = \frac{(st + end)}{2}$

(arr[mid] == key)  
return mid;

(arr[mid] < key)  
*2nd half*  $\rightarrow st = mid + 1$

(arr[mid] > key)  
*1st half*  $\rightarrow end = mid - 1$

APNA COLLEGE

09. Binary Search.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Binary Search**

Pseudocode

```
int st = 0, end = n-1;
while (st <= end) {
    mid =  $\frac{(st + end)}{2}$ 
    if (arr[mid] == key)
        return mid
    else if (arr[mid] < key) // 2nd half
        st = mid + 1
    else // 1st half
        end = mid - 1
}
```

APNA COLLEGE

11. Binary Search (Time Complexity).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Binary Search

### Time Complexity

1st itr       $\text{size} = n = \frac{n}{2^0}$

2nd itr       $n/2 = \frac{n}{2^1}$

3rd itr       $n/4 = \frac{n}{2^2}$

4th itr       $n/8 = \frac{n}{2^3}$

⋮      ⋮

TC: relation b/w inputsize & no. of op's (n)

APNA COLLEGE Loop

11. Binary Search (Time Complexity).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Binary Search

### Time Complexity

1st itr       $\text{size} = n = \frac{n}{2^0}$

2nd itr       $n/2 = \frac{n}{2^1}$

3rd itr       $n/4 = \frac{n}{2^2}$

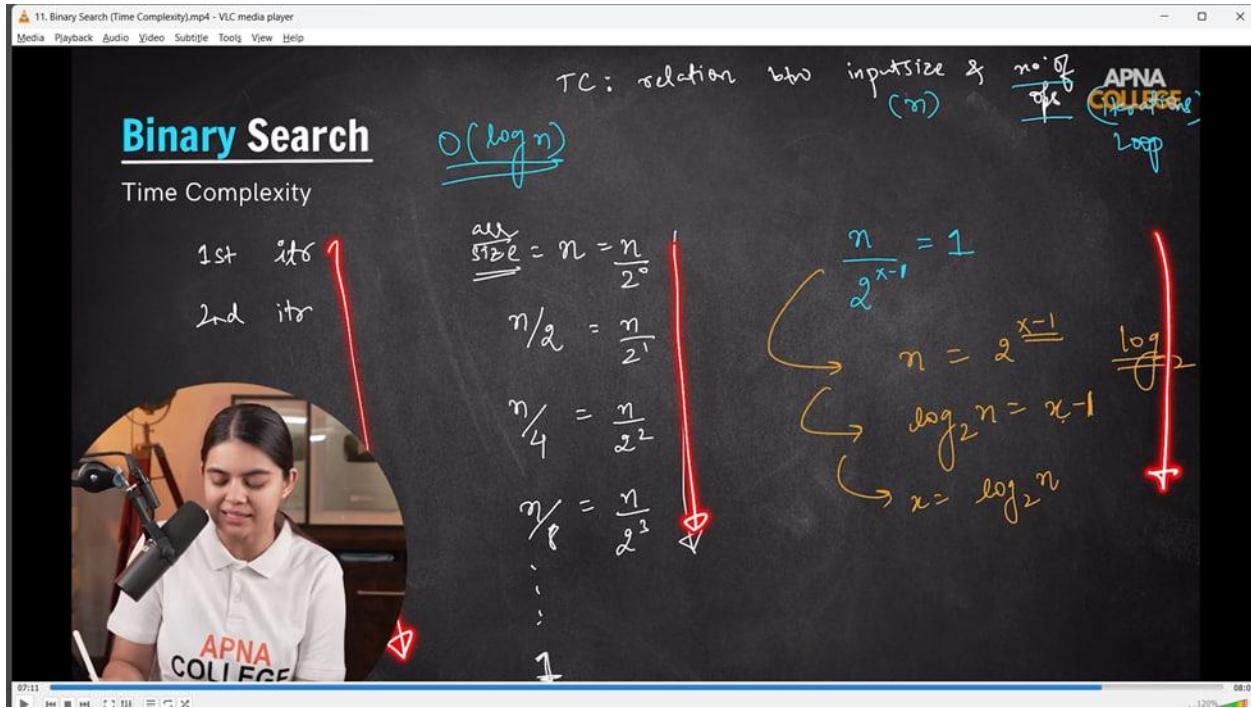
4th itr       $n/8 = \frac{n}{2^3}$

⋮      ⋮

O(log n)

TC: relation b/w inputsize & no. of op's (n)

APNA COLLEGE Loop



```

// int main()
// {
//     int n;
//     cout << "value of array size" << endl;
//     cin >> n;

//     int arr[n];
//     cout << "Write down the array elements - " << endl;
//     for (int i = 0; i < n; i++)
//     {
//         cin >> arr[i];
//     }
//     cout << "So, the inserted array is - " << endl;
//     for (int i = 0; i < n; i++)

```

```
// {
//     cout << arr[i] << " ";
// }
// cout << endl;

// int key;
// cout << "Now, enter the key you want to search for - " << endl;
// cin >> key;
// // For Binary Search Approach -
// int start = 0, end = n - 1;

// while (start <= end)
// {
//     int mid = (start + end) / 2;

//     if (arr[mid] == key)
//     {
//         cout<<"Element is here and got at - "<<mid<<endl;
//         return 0;
//     }
//     else if (arr[mid] < key)
//     {
//         start = mid + 1;
//     }
//     else if (arr[mid] > key)
//     {
```

```

//      end = mid - 1;

//      }

//      cout << "Element not found" << endl;

//  }

//  return 0;

// T.c -O(log n)

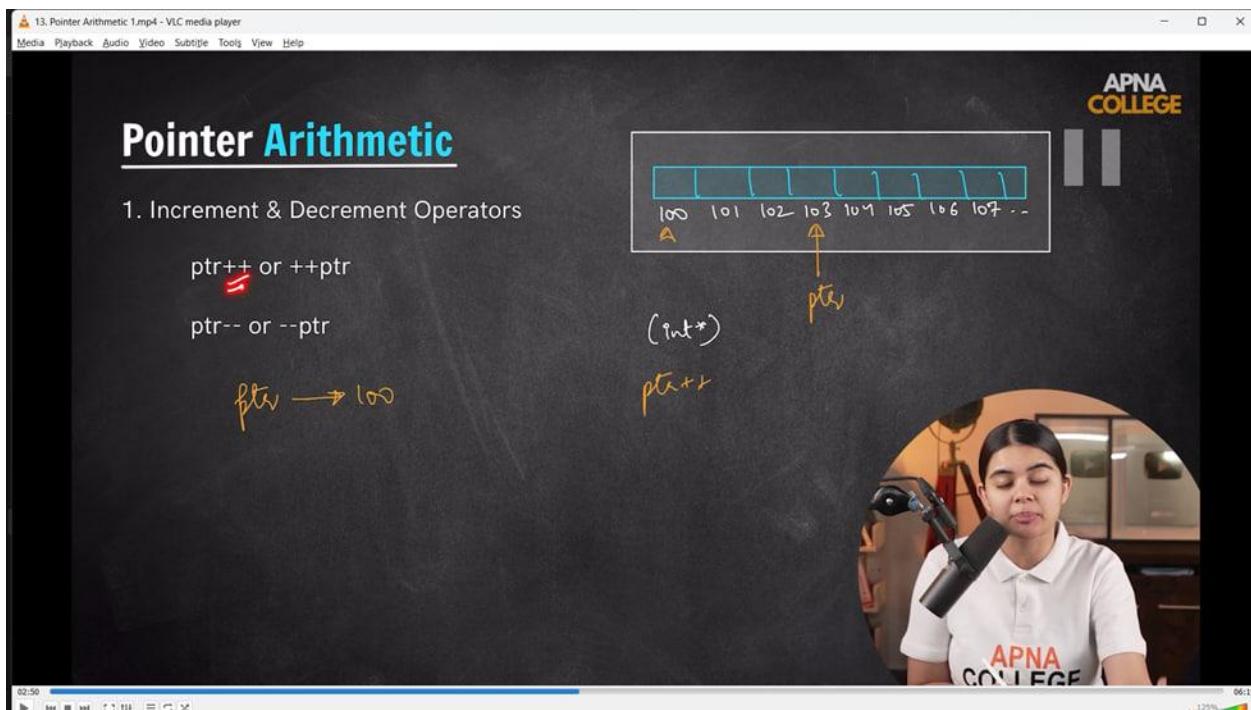
// }

// _____

```

**//7) ArrayPointer - Array is always treat as constant pointer, array pointer can;t be modified as in another variable poinyer**

### //7.1. Pointer Arithmetic Approaches -



```

// int main()

// {

```

```
// int x = 10;  
// int *ptr = &x;  
  
// int y=25;  
// ptr = &y;  
// cout<<*ptr<<"\n";//25 - in variable it is possible to update the pointer with another value  
and variable  
// return 0;  
  
// // int arr[5];  
// // cout<<arr<<endl;  
// // int y = 25;  
// // arr = &y; // showing error as - Expression must be a modifiable value  
// _____  
// //1. using increment decrement operator. In array using pointer on incrementing and  
decrementing it does the operations by the datatype size.
```

```
// int a = 10;  
// int *ptr = &a;  
// cout<<ptr<<endl;//0x61ff08  
// ptr++;  
// cout<<ptr<<endl;//0x61ff0c  
// ptr--;  
// cout<<ptr<<endl;// 0x61ff08  
// /*  
// 0x61ff08 - address of var pointer a
```

```
// 0x61ff0c - on incrementing by ++ it'll increase by 4 bytes as int size is 4 bytes, so the new address is increased.
```

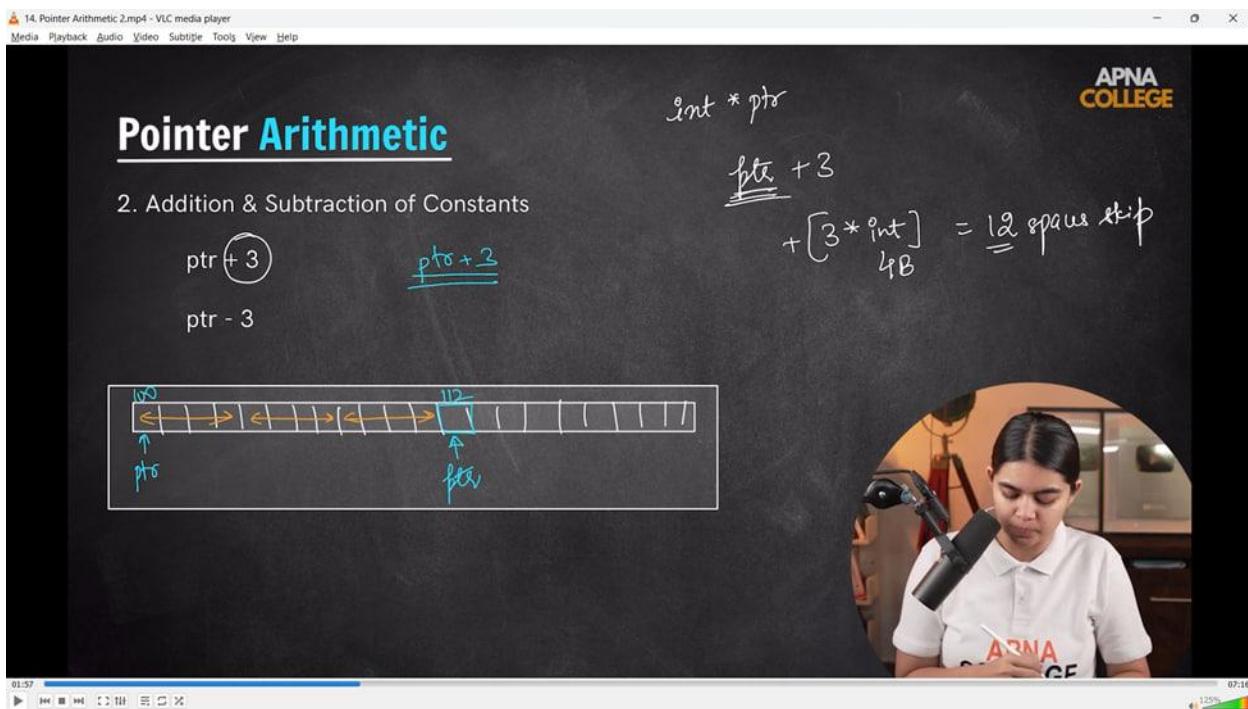
```
// In hexadecimal initially it was 08, but now it's 0c - which is 12
```

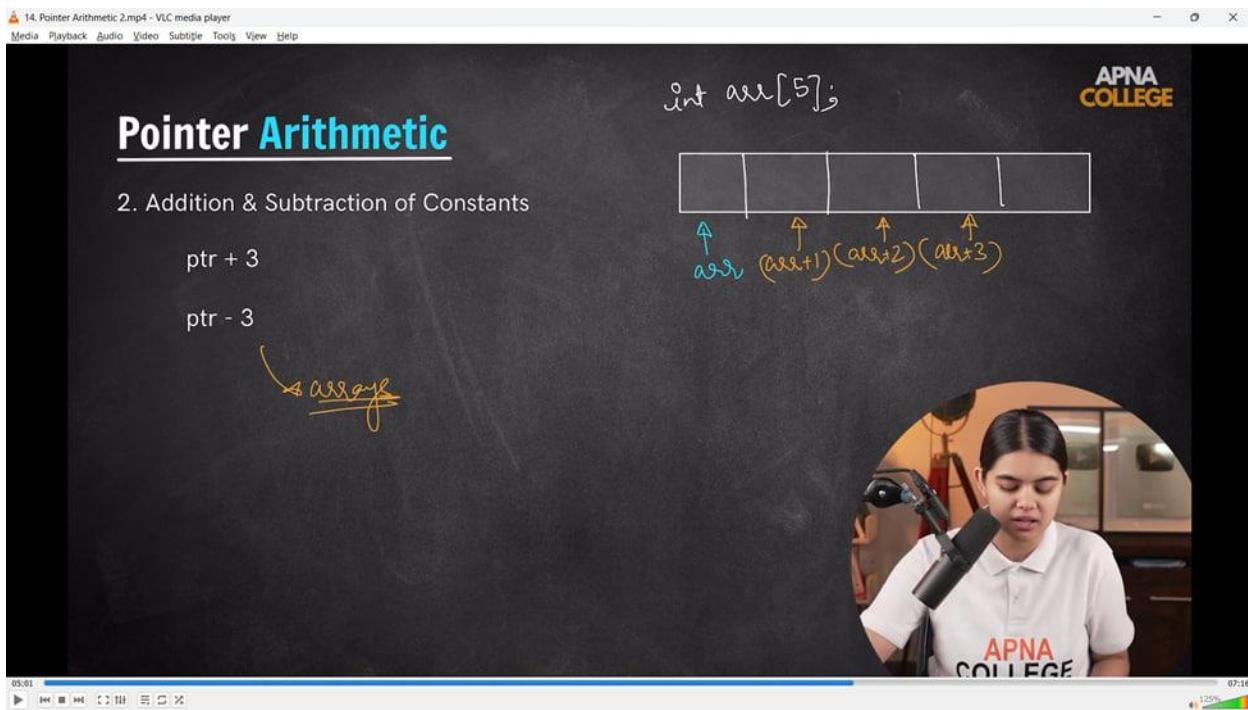
```
// 0x61ff08 - on doing the decrement operation, address again decreased by 4 bytes.
```

```
// */
```

```
// _____
```

### /// 7.2 - Addition & Subtraction of Constants -





/// 2. Addition & Subtraction of Constants -

```
// int a1 = 5;
// int *ptr1 = &a1;
// cout<<ptr1<<endl;//0x61ff00 - here address is 0
// cout<<(ptr1+3)<<endl;//0x61ff0c - here increased by3*4bytes = 12 bytes which is c.
/// Pointer me addition ya substr. krne pr int ke size ke terms me hota he. Yaha add 3 * 4 byte as
size of int = 12 bytes
// cout<<(ptr1-3)<<endl;
// }
```

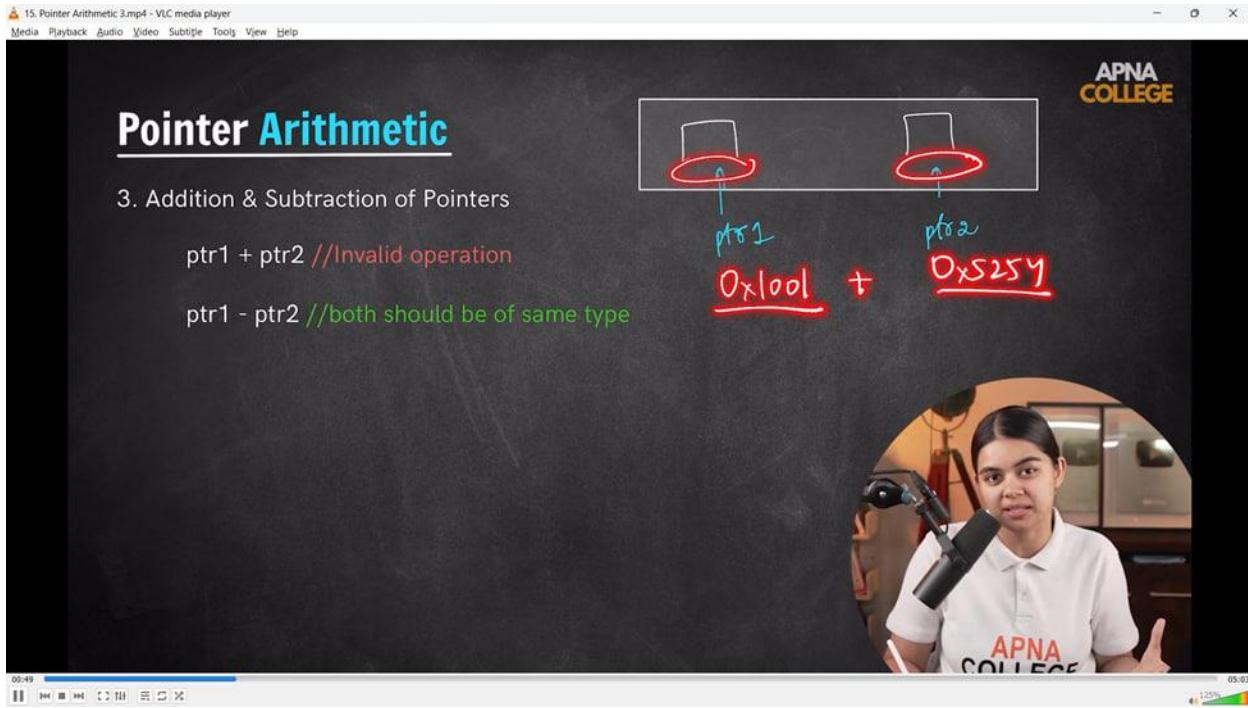
// interms of array by Arithmatic operation using Dereferenc eoperator -

```
// void printArr(int *ptr, int n)
//{
//    for (int i = 0; i < 7; i++)
```

```
// {
//     cout << *(ptr + i) << endl;
// }
// }

// int main()
// {
//     int arr[] = {2, 4, 5, 6, 10, 12, 17};
//     int n = sizeof(arr) / sizeof(int);
//     printArr(arr, n);
// }

// /*
// 2
// 4
// 5
// 6
// 10
// 12
// 17
// so here printing all array values using dereference operator */
// _____
// 7.3- Addition & Subtraction of Pointers
```



```
int main()
{
    int a = 5;
    int *ptr1 = &a;
    int *ptr2 = ptr1 + 3;
```

```
cout<<ptr1<<endl;//0x61ff04
cout<<ptr2<<endl;//0x61ff10
cout<<ptr2 - ptr1 <<endl;//3
```

```
// in terms of array -
int arr[20] = {1,2,3,4,5,6};
int *ptr3 = arr; // base address of array
int *ptr4 = ptr3+3;// base add + 3bytes
```

```
cout<<*ptr3<<endl;//1
```

```
cout<<*ptr4<<endl;//4
```

#### // 7.4 - Comparision Operators

```
cout<<(ptr2>ptr1)<<endl;//1 - Yes : True
```

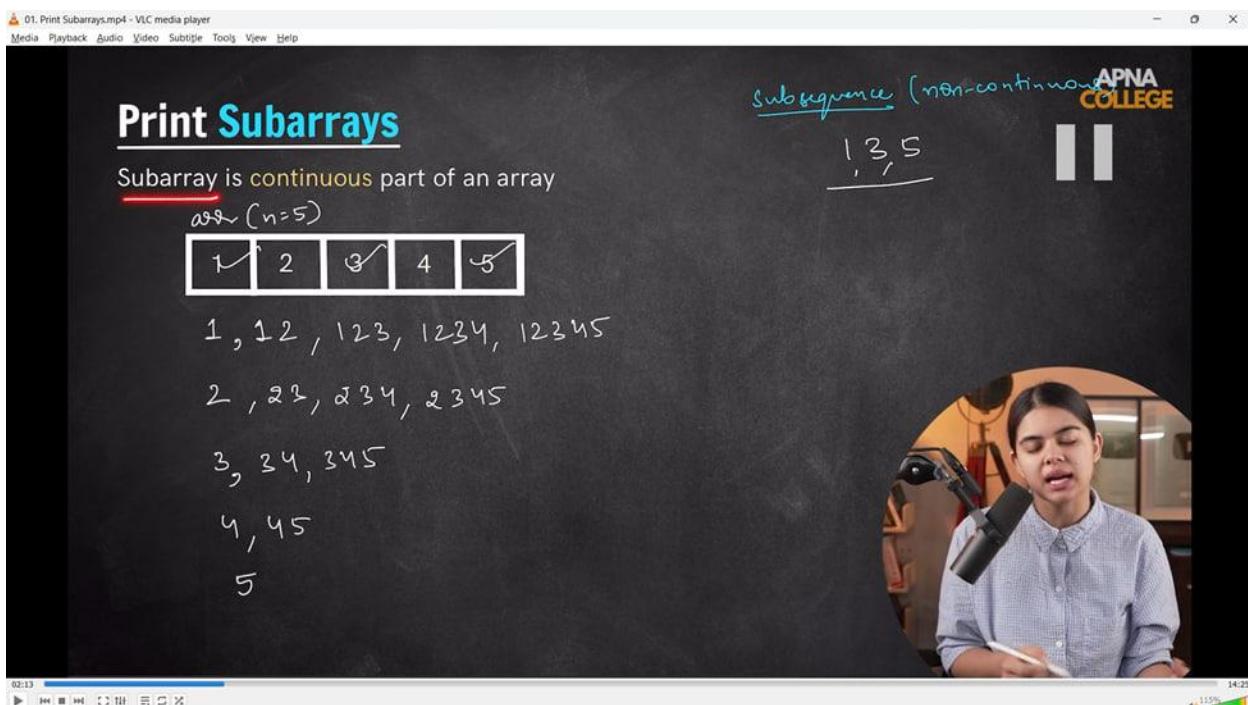
```
cout<<(ptr4<ptr3)<<endl;// 0 - NO : False
```

```
cout<<(ptr3 == arr)<<endl;//1 - YES ptr3 and arr are same pointing
```

```
}
```

```
// _____
```

#### //8) Subarrays -



01. Print Subarrays.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Print Subarrays

Subarray is continuous part of an array

$\text{arr}(n=5)$

1	2	3	4	5
---	---	---	---	---

$1, 12, 123, 1234, 12345 \rightarrow 5$

$2, 23, 234, 2345 \rightarrow 4$

$3, 34, 345 \rightarrow 3$

$4, 45 \rightarrow 2$

$5 \rightarrow 1$

$m \text{ size}$

$$m=5 \quad 5+4+3+2+1 = 15 = \frac{5*6}{2} = 15$$

$$m=4 \quad 4+3+2+1 = 10 = \frac{5*4}{2} = 10$$

$$m \quad \frac{m+(m-1)+(m-2)+\dots+1}{2} = \frac{m+(m+1)}{2}$$

APNA COLLEGE

01. Print Subarrays.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Print Subarrays

Subarray is continuous part of an array

$\text{arr}(n=5)$

1	2	3	4	5
---	---	---	---	---

$1, 12, 123, 1234, 12345 \rightarrow 5$

$2, 23, 234, 2345 \rightarrow 4$

$3, 34, 345 \rightarrow 3$

$4, 45 \rightarrow 2$

$5 \rightarrow 1$

$\text{start}$        $\text{end}$        $\text{start}$        $\text{end}$

0	0, 1, 2, 3, 4	( $\text{start} \rightarrow n$ )
1	1, 2, 3, 4	( $\text{start} \rightarrow n-1$ )
2	2, 3, 4	( $\text{start} \rightarrow n-2$ )
3	3, 4	
4	4	

APNA COLLEGE

```
// void printSubArray(int Prices[], int n)

// {
//   cout << "So, pairwise subarrays are - " << endl;
```

```
// for (int i = 0; i < n; i++)
// {
//     for (int j = i; j < n; j++)
//     {
//         cout << "(" << Prices[i] << "," << Prices[j] << ")";
//     }
//     cout << endl;
// }
// }
```

```
// void SubArray2(int Prices[], int n)
//{
//    cout << "and the subarrays are - " << endl;
//    for (int i = 0; i < n; i++)
//    {
//        for (int j = i; j < n; j++)
//        {
//            cout << Prices[j] << " ";
//        }
//        cout << endl;
//    }
//}
```

//// OR

```
// int main()
//{
//    int n;
```

```
// cout << "value of array size" << endl;
// cin >> n;

// int Prices[n];
// cout << "Write down the array elemenets - " << endl;
// for (int i = 0; i < n; i++)
// {
//     cin >> Prices[i];
// }
// cout << "So, the inserted array is - " << endl;
// for (int i = 0; i < n; i++)
// {
//     cout << Prices[i] << " ";
// }
// cout << endl;

// printSubArray(Prices, n);
// SubArray2(Prices, n);
// /*
// value of array size
// 5
// Write down the array elemenets -
// 12 23 56 45 89
// So, the inserted array is -
// 12 23 56 45 89
// So, pairwise subarrays are -
```

```
// (12,12)(12,23)(12,56)(12,45)(12,89)
// (23,23)(23,56)(23,45)(23,89)
// (56,56)(56,45)(56,89)
// (45,45)(45,89)
// (89,89)

// and the subarrays are -
// 12 23 56 45 89
// 23 56 45 89
// 56 45 89
// 45 89
// 89

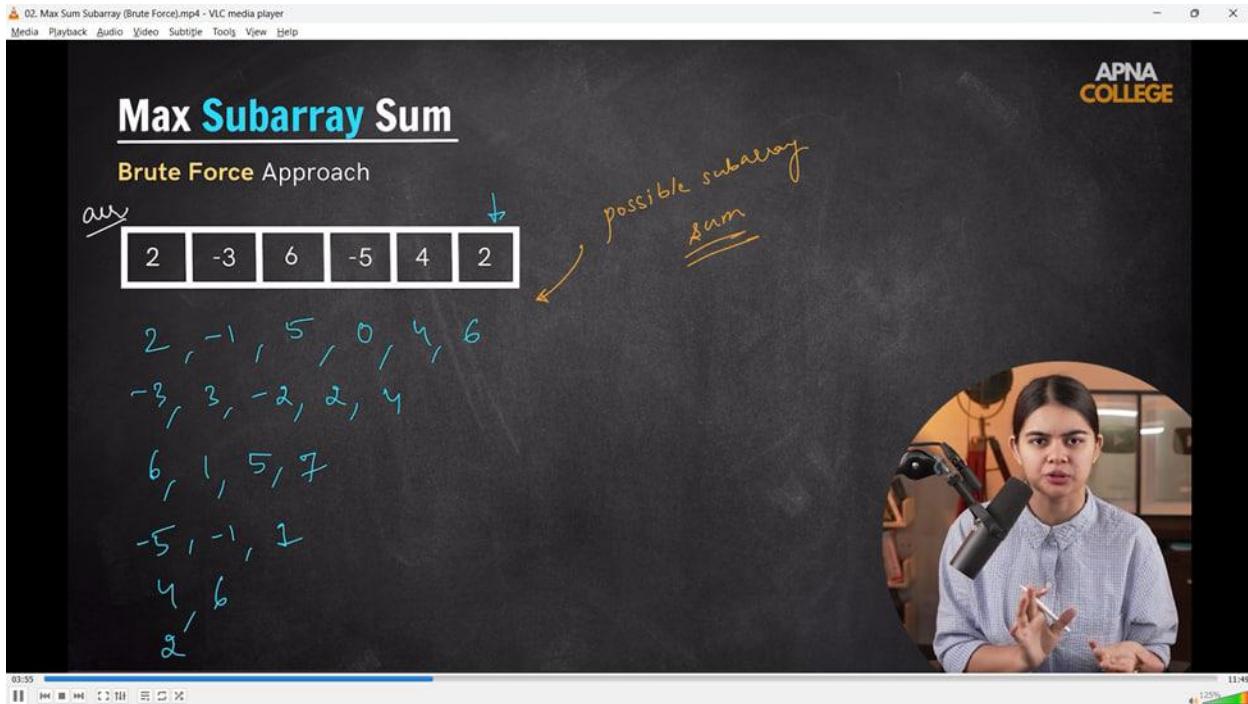
// T.C - O(n^3)

// */
// }

// _____
```

**// 9) Max Subarray Sum -**

**// 9.1 - Clearly Brute Force Approach -**



```
// void maxSubArraySum(int Prices[], int n)

// {
//     int maxSum = INT_MIN;
//     cout << "Noe for maximum subarray sum - " << endl;
//     // in form of subarray the sum
//     for (int i = 0; i < n; i++)
//     {
//         for (int j = i; j < n; j++)
//         {
//             int currSum = 0;
//             for (int k = i; k <= j; k++)
//             {

```

```

// currSum += Prices[k]; // for prnitng the current sum of every subarray possible from
every element

// }

// cout << currSum << ",";

// maxSum = max(maxSum, currSum);

// }

// cout << endl;

// }

// cout << "Maximum Subarray sum is - " << maxSum << endl;

// }

// int main()

// {

// int n;

// cout << "value of array size" << endl;

// cin >> n;

// int Prices[n];

// cout << "Write down the array elemenets - " << endl;

// for (int i = 0; i < n; i++)

// {

// cin >> Prices[i];

// }

// cout << "So, the inserted array is - " << endl;

// for (int i = 0; i < n; i++)

// {

```

```
//      cout << Prices[i] << " ";
//  }

//  cout << endl;

//  maxSubArraySum(Prices, n);

//  /*
//   value of array size
//   6
//   Write down the array elemenets -
//   2 -3 6 -5 4 2
//   So, the inserted array is -
//   2 -3 6 -5 4 2
//   Note for maximum subarray sum -
//   2,-1,5,0,4,6,
//   -3,3,-2,2,4,
//   6,1,5,7,
//   -5,-1,1,
//   4,6,
//   2,
//   Maximum Subarray sum is - 7

//   T.C - O(n^3)
//   */
// }
```

## // 9.2 - SLightly Optimizede approch - for decreasing the time complexity

```
// void maxSubArraySum2(int Prices[], int n)

// {
//     int maxSum = INT_MIN;
//     for (int i = 0; i < n; i++)
//     {
//         int currSum = 0;
//         for (int j = i; j < n; j++)
//         {
//             currSum += Prices[j];
//             maxSum = max(maxSum, currSum);
//         }
//         cout << endl;
//     }
//     cout << "Maximum Subarray sum is - " << maxSum << endl;
// }

// int main()

// {
//     int n;
//     cout << "value of array size" << endl;
//     cin >> n;

//     int Prices[n];
//     cout << "Write down the array elemenets - " << endl;
//     for (int i = 0; i < n; i++)
```

```
// {
//     cin >> Prices[i];
// }
// cout << "So, the inserted array is - " << endl;
// for (int i = 0; i < n; i++)
// {
//     cout << Prices[i] << " ";
// }
// cout << endl;
```

```
// // maxSubArraySum(Prices, n);
// maxSubArraySum2(Prices, n);
```

```
// /*
// value of array size
// 6
// Write down the array elemenets -
// 2 -3 6 -5 4 2
// So, the inserted array is -
// 2 -3 6 -5 4 2
```

```
// Maximum Subarray sum is - 7
// T.C - O(n^2)
// */
// }
```

## // 9.3 - Using Most Optimized Approach - Kadane's Algorithms. For very less Time Complexity

A screenshot of a VLC media player window displaying a video from APNA COLLEGE. The video shows a chalkboard with the title "Max Subarray Sum" and "Kadane's Algorithm (DP)". Below the title is a horizontal array of numbers: 2, -3, 6, -5, 4, 2. A red bracket under the array indicates the current subarray being considered. Above the array, a yellow arrow points down to the first element, and another yellow arrow points up to the last element. To the right of the array, handwritten code in blue and green is shown:

```
currSum = 0  
maxSum = INT_MIN  
  
for (int i=0 to n)  
    currSum += arr[i]  
    maxSum = max(maxSum, currSum)  
    if (currSum < 0)  
        currSum = 0
```

The video player interface at the bottom shows a progress bar from 03:53 to 07:07.

A screenshot of a VLC media player window displaying a video from APNA COLLEGE. The video shows a chalkboard with the title "Max Subarray Sum" and "Kadane's Algorithm (DP)". Below the title is a horizontal array of numbers: 2, -3, 6, -5, 4, 2. A red bracket under the array indicates the current subarray being considered. Above the array, five yellow arrows point up to each element in the array. To the right of the array, handwritten code in blue and green is shown, identical to the one in the previous frame. Above the code, there is a note: "currSum = 0x -108.257" and "maxSum = INT\_MIN 2.8.7". A red circle highlights the number "7" in "2.8.7", and a red checkmark is placed next to it. The video player interface at the bottom shows a progress bar from 05:31 to 07:07.

```
// void maxSubArraySum3(int Prices[], int n)  
// {
```

```
// int maxSUM = INT_MIN;  
  
// int currSum = 0;  
  
// for (int i = 0; i < n; i++)  
// {  
//     currSum += Prices[i];  
//     maxSUM = max(maxSUM, currSum);  
//     if (currSum < 0)  
//     {  
//         currSum = 0;  
//     }  
// }  
  
// cout << "Maximum Subarray sum is - " << maxSUM << endl;  
//}  
  
// int main()  
//{  
//    int n;  
//    cout << "value of array size" << endl;  
//    cin >> n;  
  
//    int Prices[n];  
//    cout << "Write down the array elements - " << endl;  
//    for (int i = 0; i < n; i++)  
//    {  
//        cin >> Prices[i];  
//    }  
//    cout << "So, the inserted array is - " << endl;
```

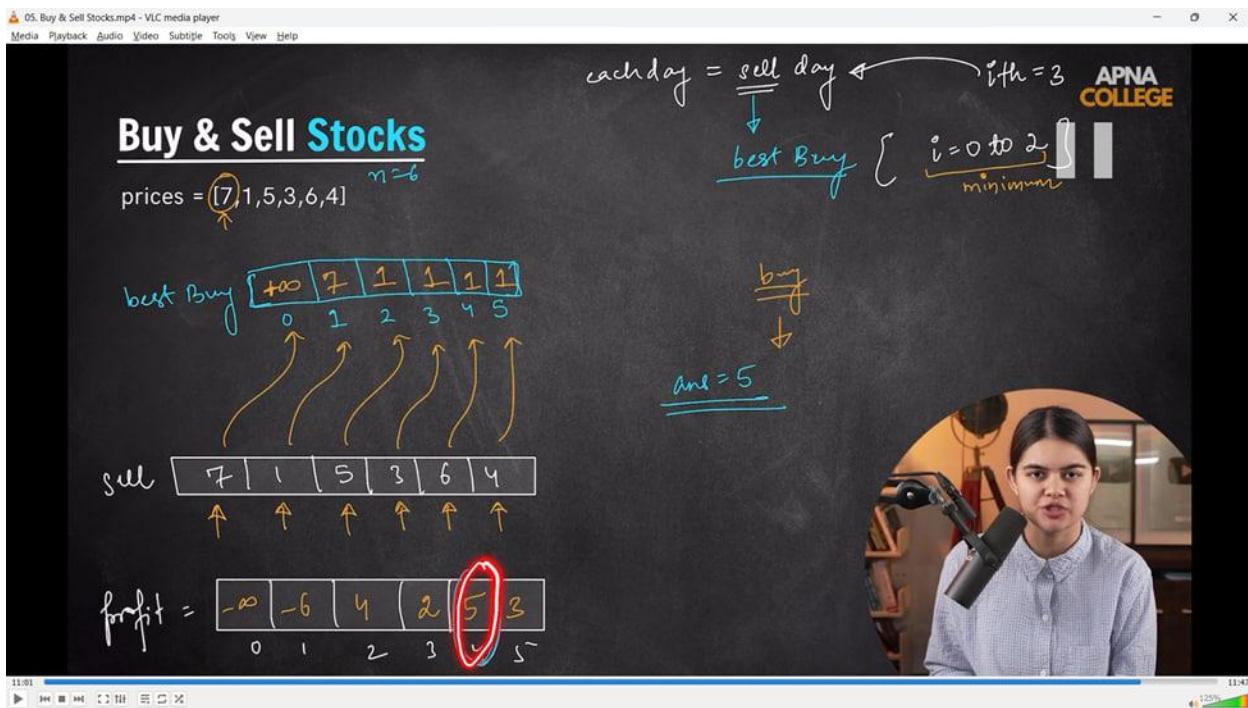
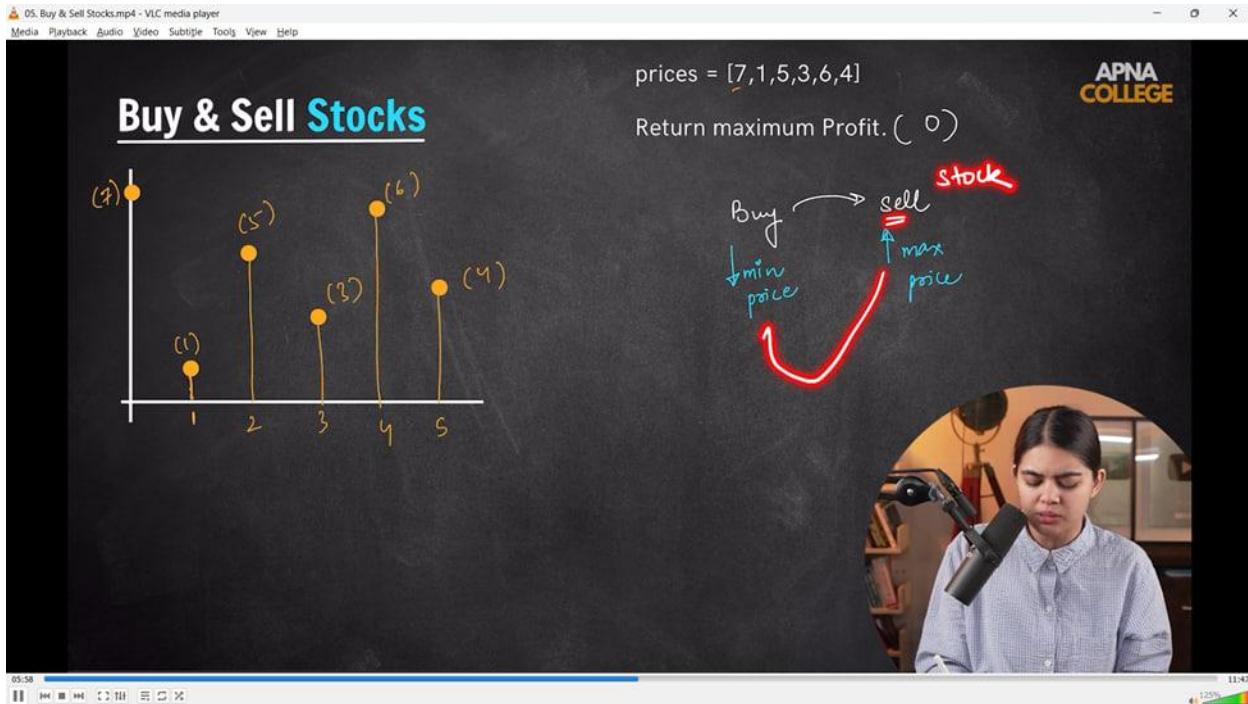
```

// for (int i = 0; i < n; i++)
//
// {
//     cout << Prices[i] << " ";
//
// }
// cout << endl;

// // maxSubArraySum(Prices, n);
// maxSubArraySum3(Prices, n);
// /*
// value of array size
// 6
// Write down the array elements -
// 2 -3 6 -5 4 2
// So, the inserted array is -
// 2 -3 6 -5 4 2
// Maximum Subarray sum is - 7
// T.C - O(n)
// */
// }

// _____
//10) bUY & sELL sTOCK pROBLEM-

```



```
// void maxprofit(int Prices[], int n)
//{

```

```
// int bestBuy[100000]; // as per the given max limits
// bestBuy[0] = INT_MAX;

// for (int i = 0; i < n; i++)
// {
//     bestBuy[i] = min(bestBuy[i - 1], Prices[i - 1]);
// }

// int maxProfit = 0;
// for (int i = 0; i < n; i++)
// {
//     int currProfit = Prices[i] - bestBuy[i];
//     maxProfit = max(maxProfit, currProfit);
// }
// cout << "MaxProfit is - " << maxProfit << endl;
// }

// int main()
// {
//     int n;
//     cout << "what is the vale of n - " << endl;
//     cin >> n;

//     int Prices[n];
//     cout << "Mentin the array elementa" << endl;
//     for (int i = 0; i < n; i++)
//     {
```

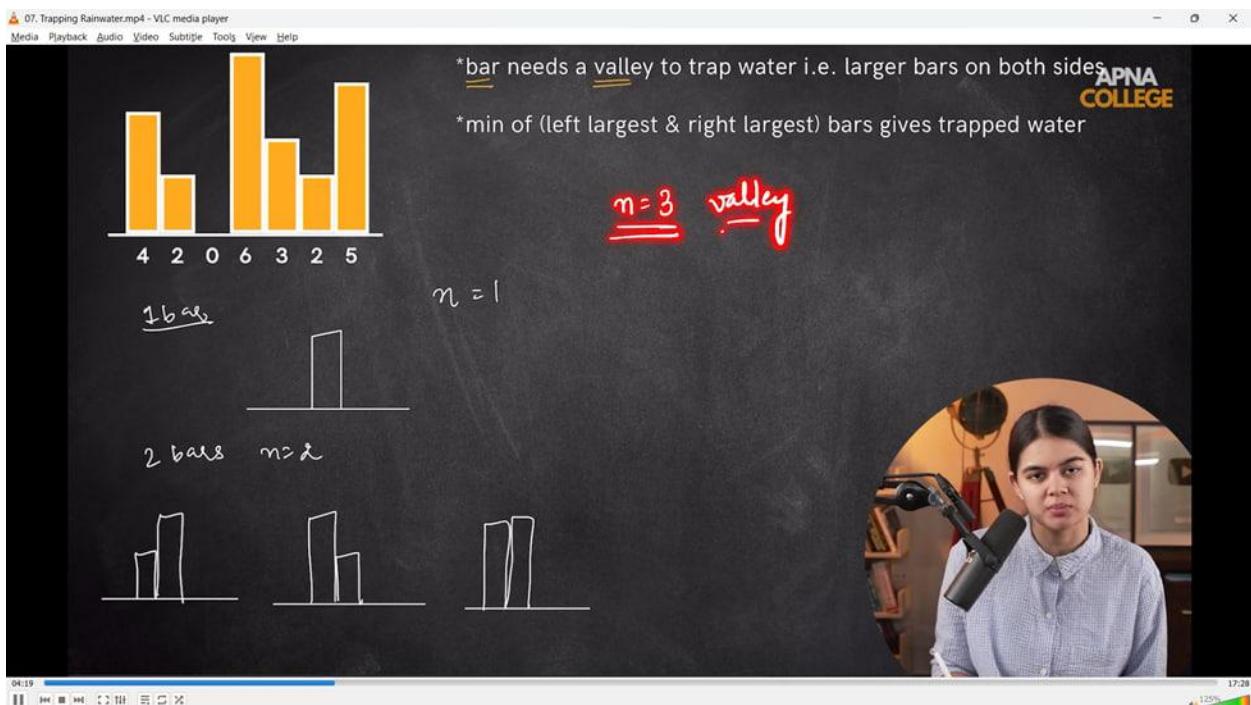
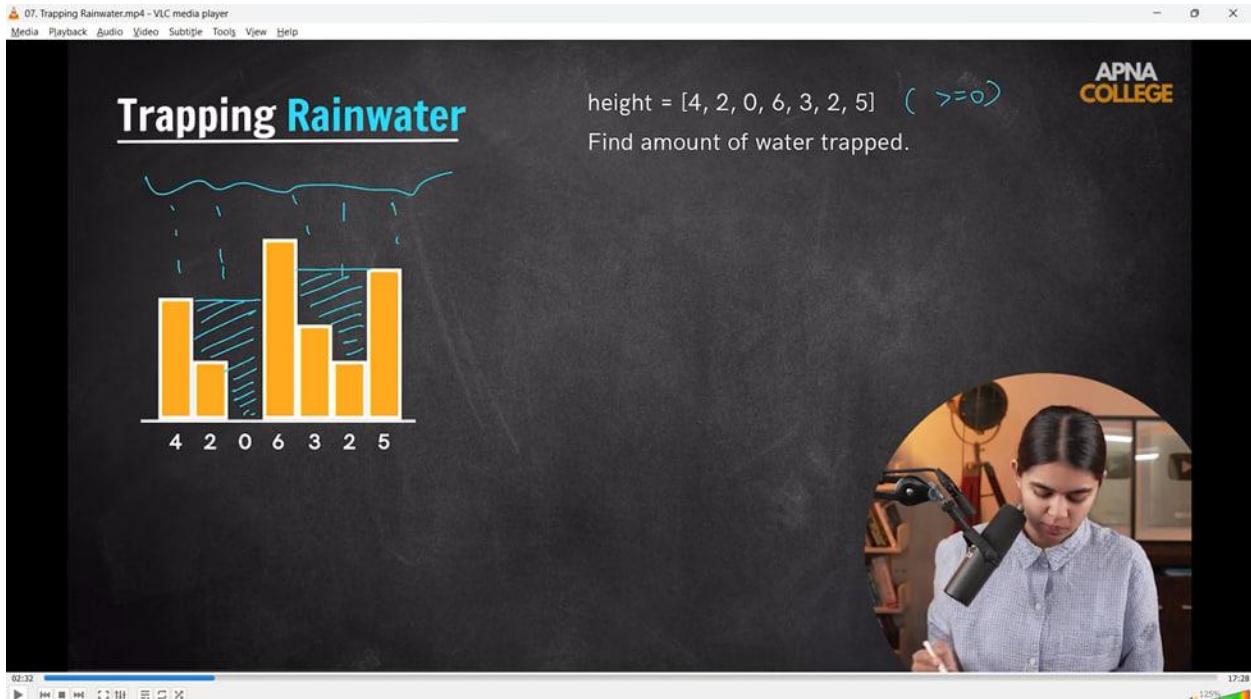
```
//      cin >> Prices[i];  
//  }  
//  cout << "So, the entered elements of array is - " << endl;  
//  for (int i = 0; i < n; i++)  
//  {  
//      cout << Prices[i] << " ";  
//  }  
//  cout << endl;  
//  maxprofit(Prices, n);
```

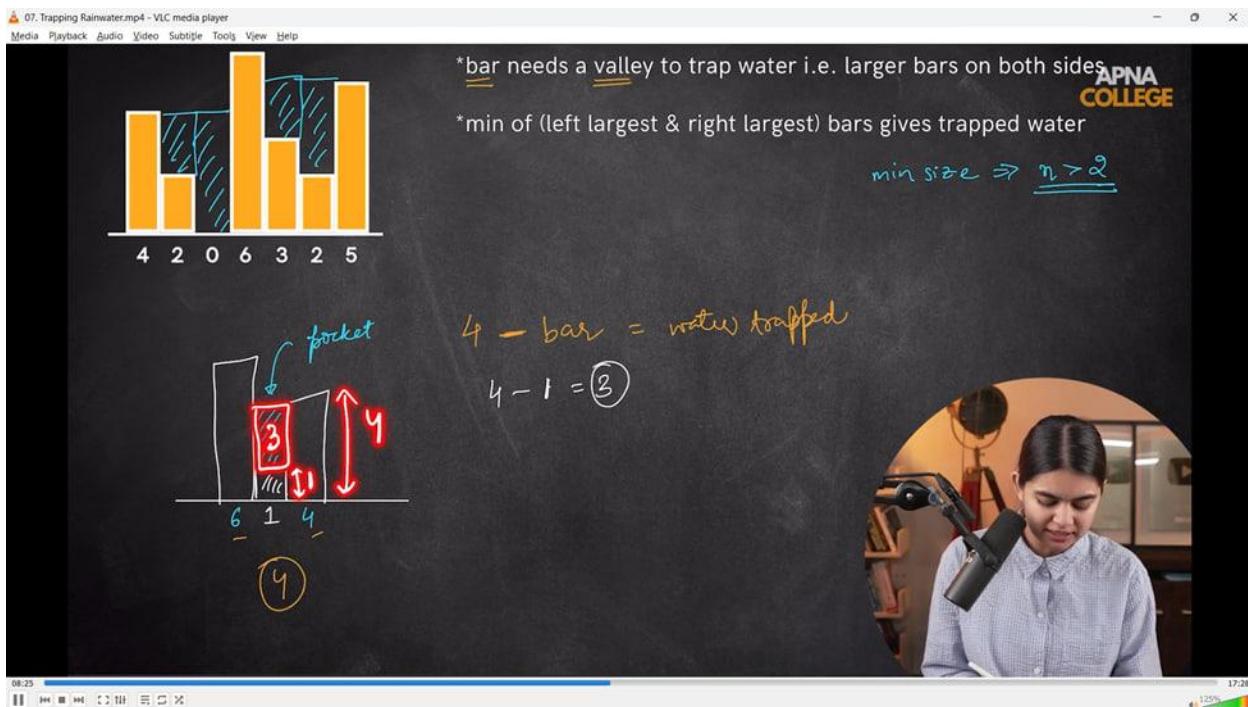
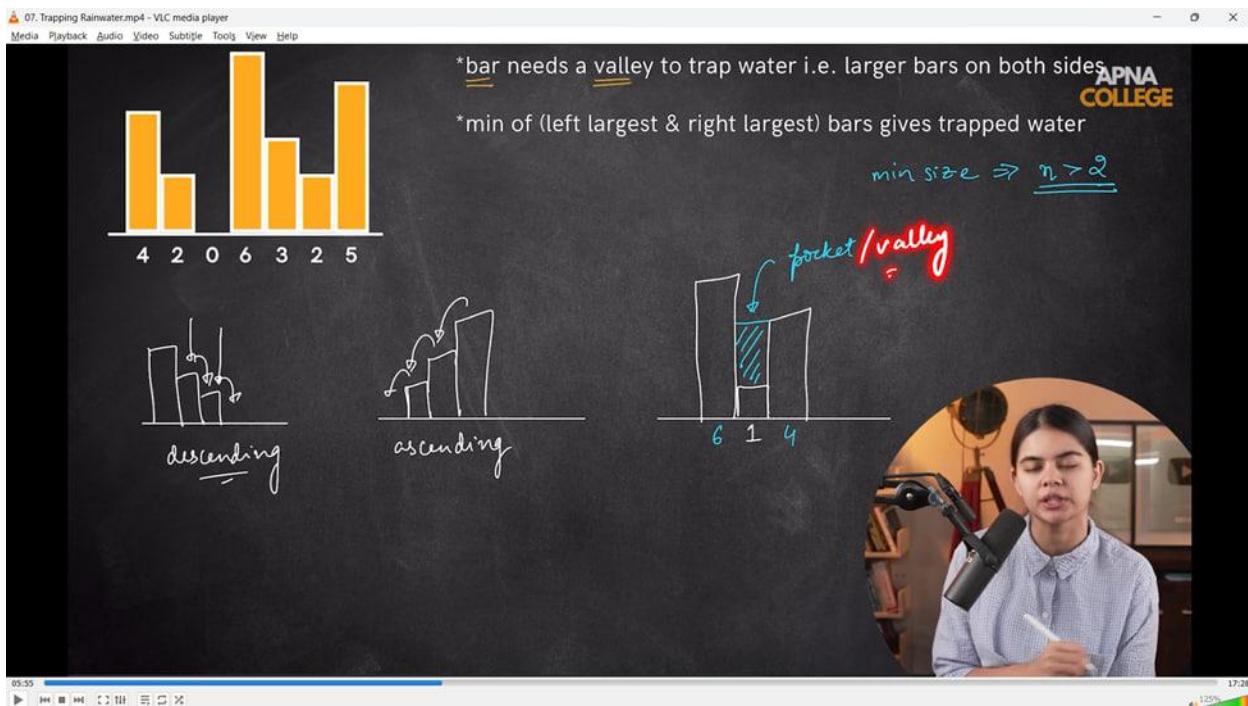
```
// /*  
// what is the vale of n -  
// 5  
// Mentin the array elementa  
// 9 5 15 8 2  
// So, the entered elements of array is -  
// 9 5 15 8 2  
// MaxProfit is - 15
```

```
// what is the vale of n -  
// 6  
// Mentin the array elementa  
// 7 1 5 3 6 4  
// So, the entered elements of array is -  
// 7 1 5 3 6 4  
// MaxProfit is - 7
```

```
// */  
//}  
//
```

## //11) Trapping Rainwater Problem -





07. Trapping Rainwater.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

\*bar needs a valley to trap water i.e. larger bars on both sides

\*min of (left largest & right largest) bars gives trapped water

min size  $\Rightarrow n > 2$

$4 - \text{bar} = \text{water trapped}$

$4 - 1 = 3$

$\text{pocket}$

$6 \quad 1 \quad 4$

(4)

08:55 17:28 125%

APNA COLLEGE

A circular video feed of a female teacher speaking into a microphone.

07. Trapping Rainwater.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

\*bar needs a valley to trap water i.e. larger bars on both sides

\*min of (left largest & right largest) bars gives trapped water

min size  $\Rightarrow n > 2$

$4 - \text{bar} = \text{water trapped}$

$4 - 1 = 3 \times 3 = 9$

$\text{height} \times \frac{\text{width}}{(1)}$

$\text{pocket}$

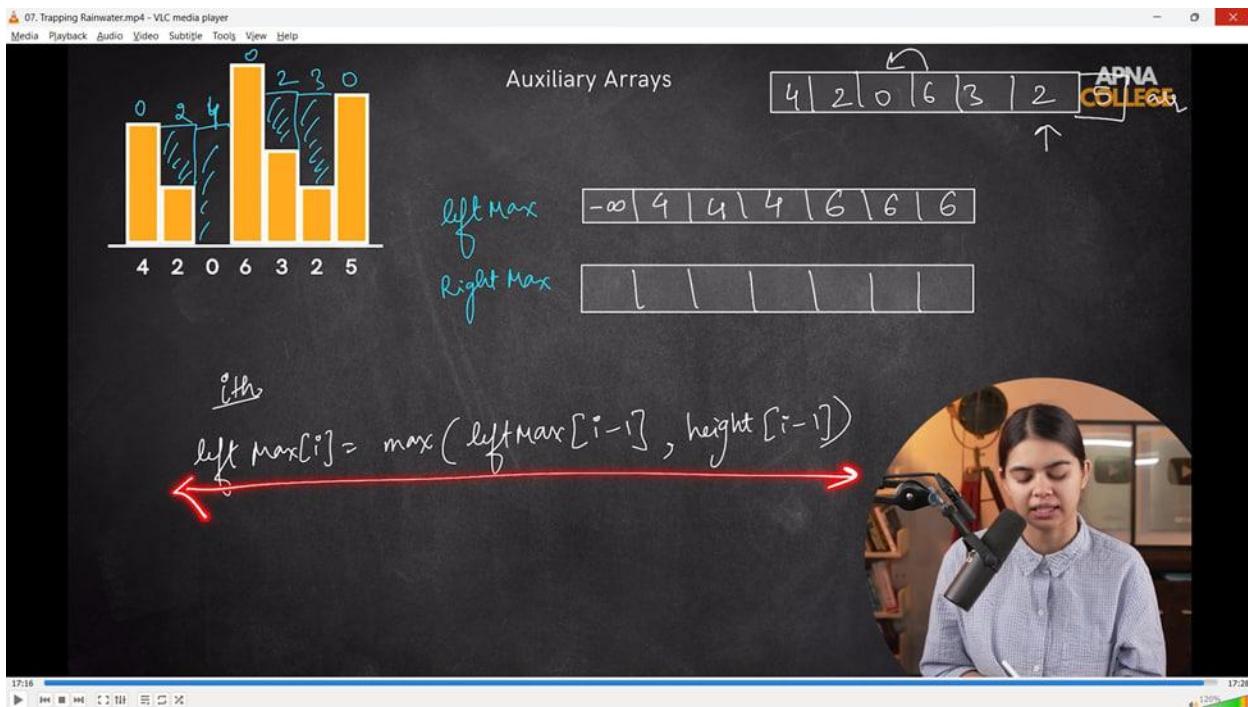
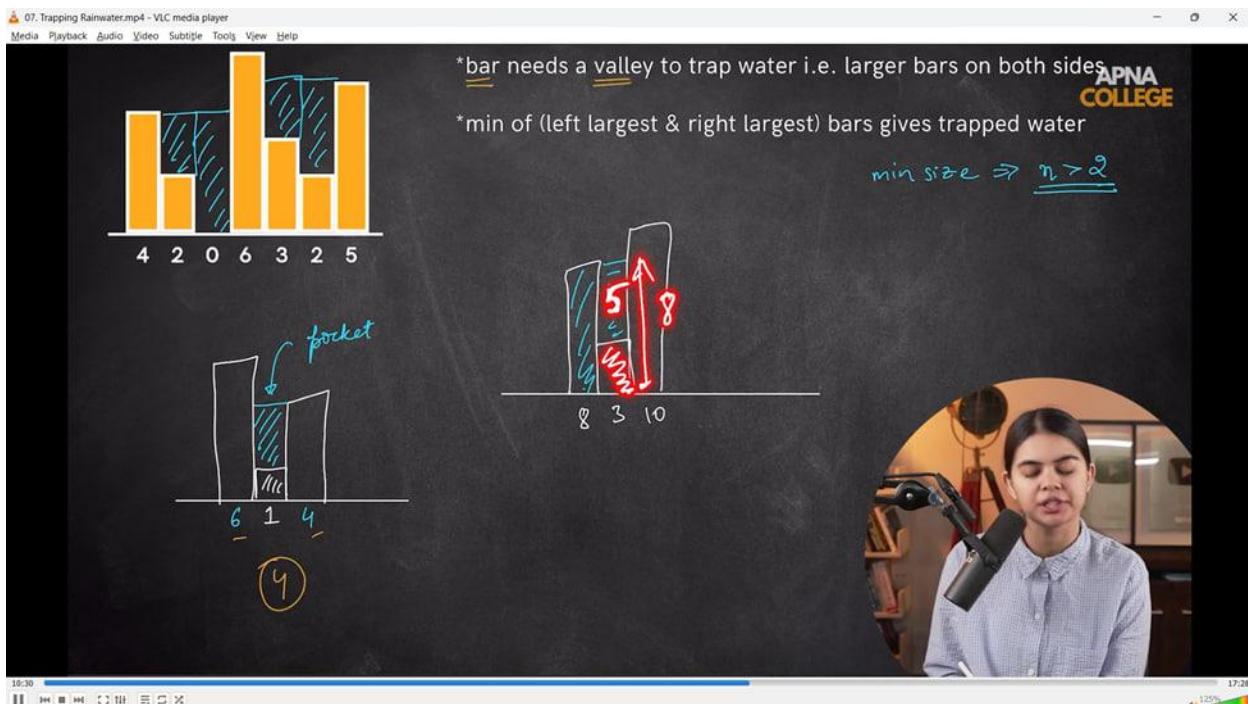
$6 \quad 1 \quad 4$

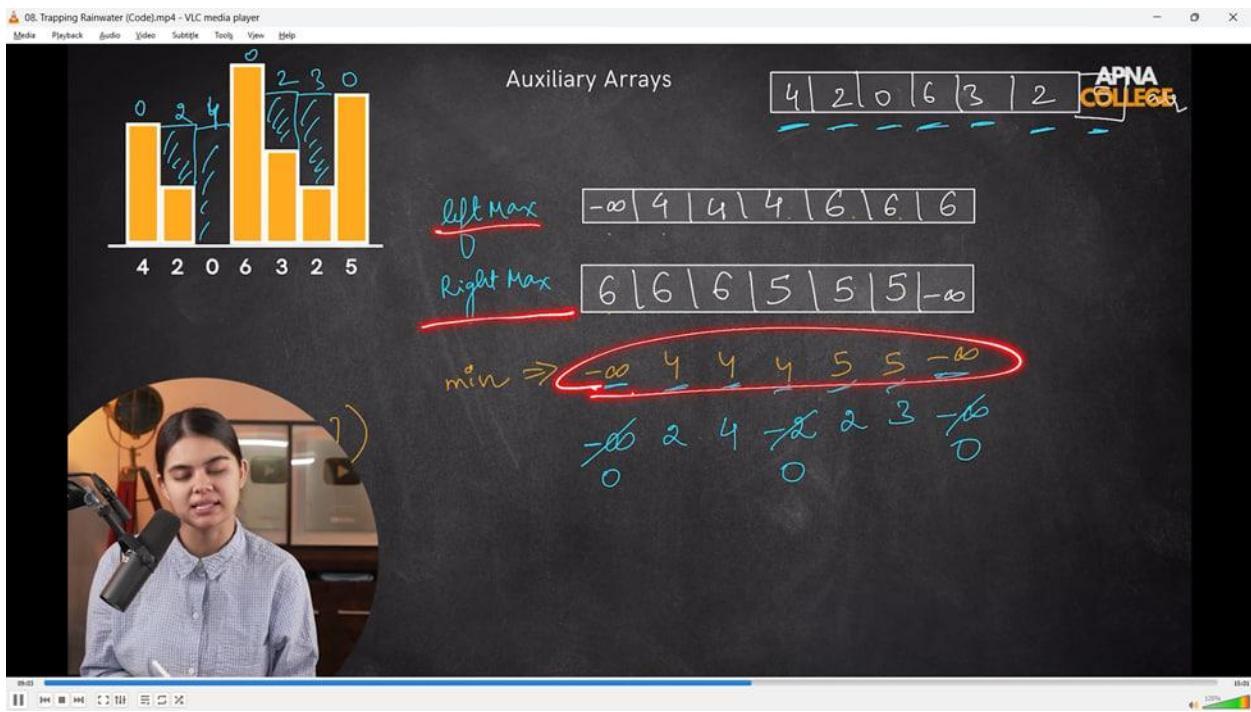
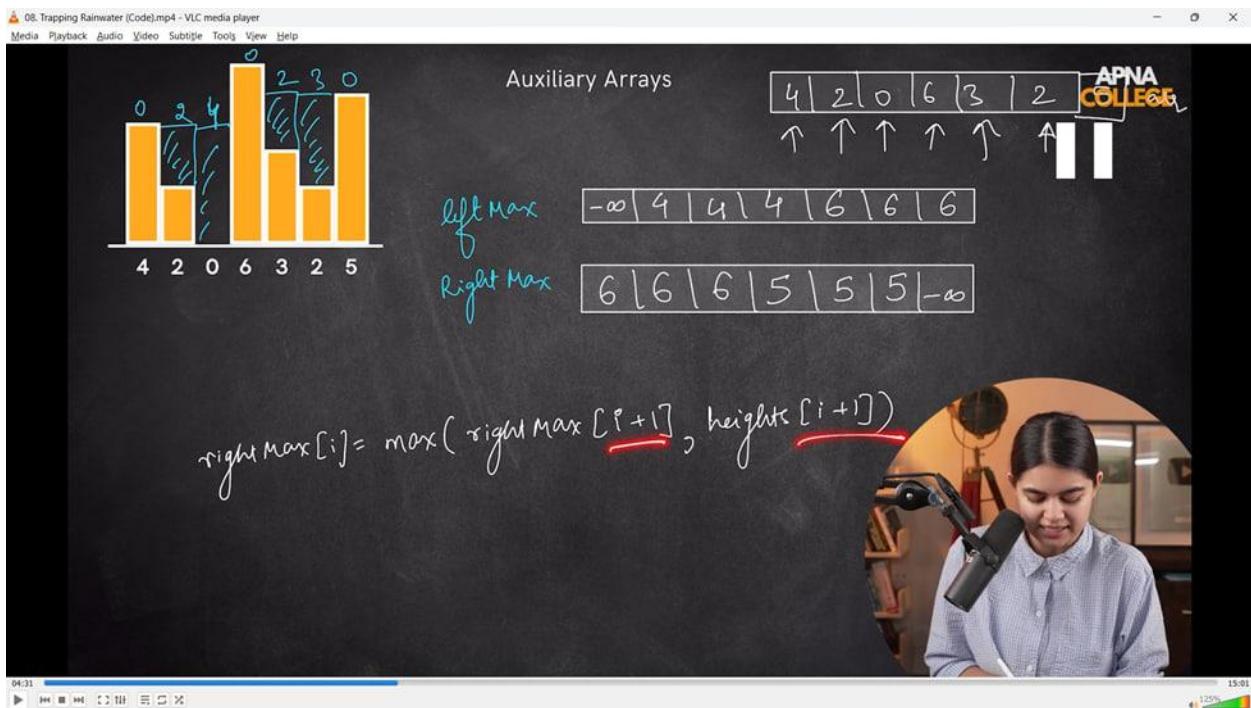
(4)

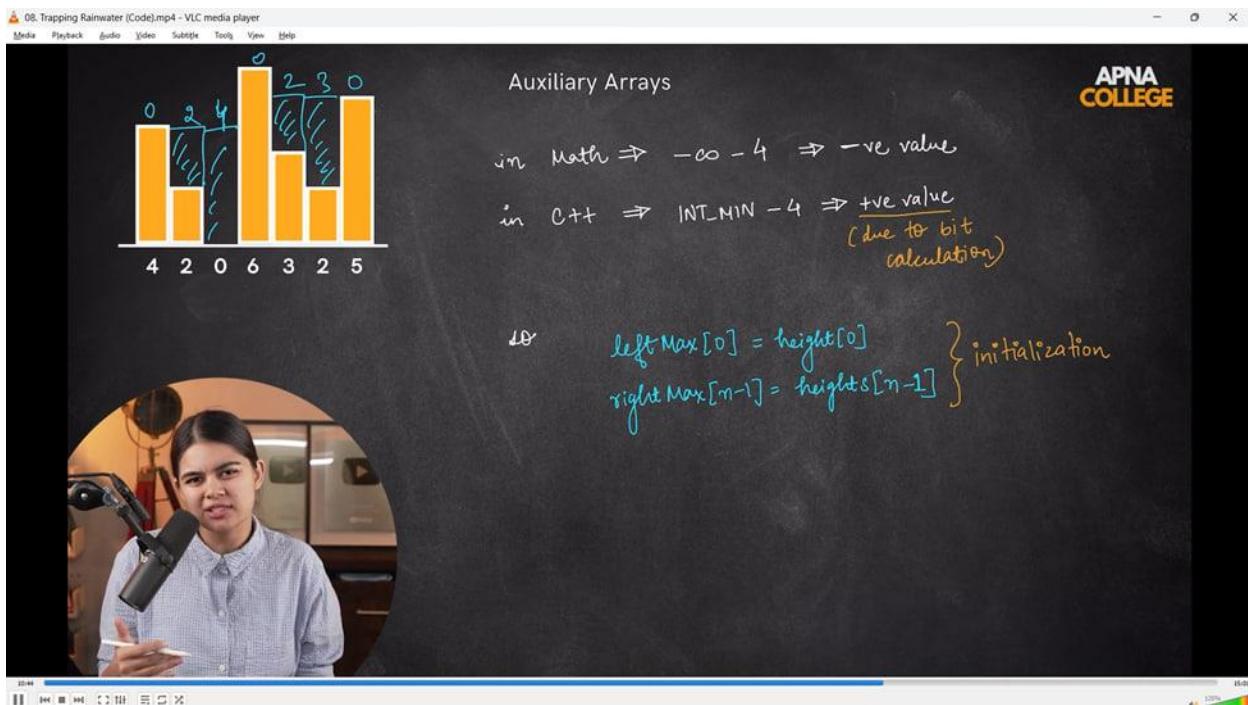
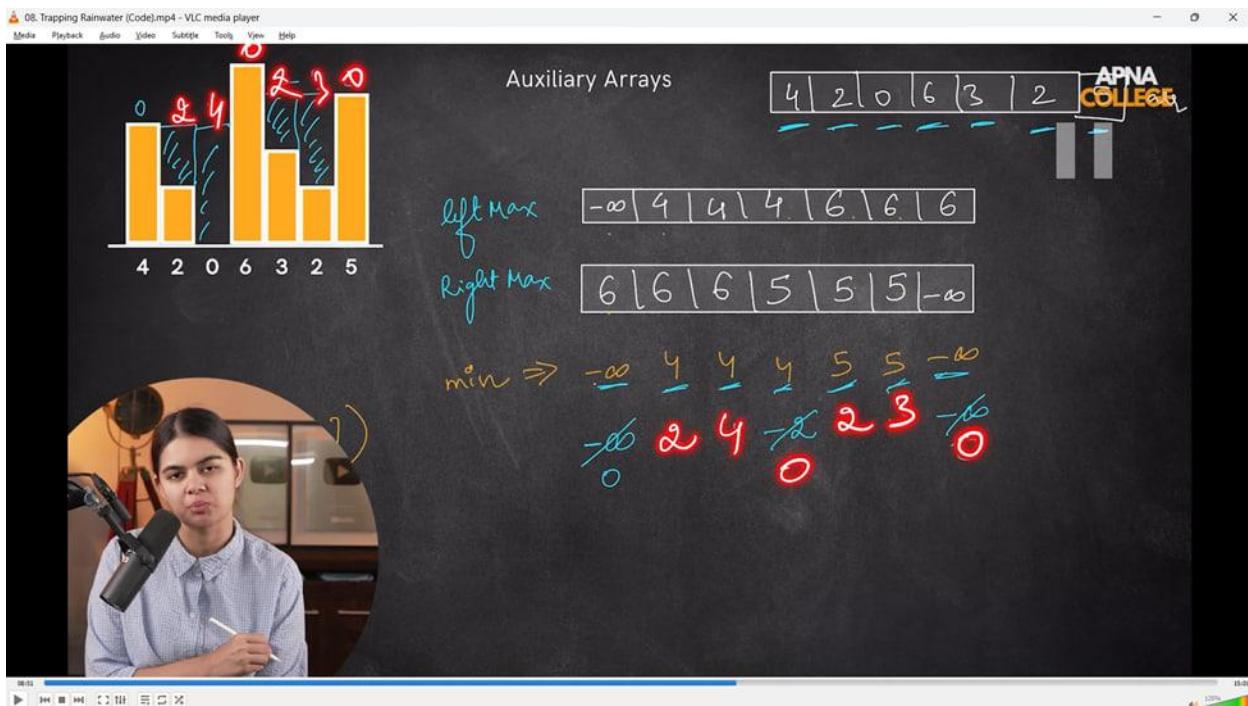
09:38 17:28 125%

APNA COLLEGE

A circular video feed of a female teacher speaking into a microphone.







```
int Trap(int heights[], int n)
{
    int leftMax[20000], rightMax[20000];
```

```

leftMax[0] = heights[0];

rightMax[n - 1] = heights[n - 1];

// For getting the left max side for every bars

cout << "So, the Left Valley max values for each bar is - " << endl;

cout << leftMax[0] << ",";

for (int i = 1; i < n; i++)

{

    leftMax[i] = max(leftMax[i - 1], heights[i - 1]);

    cout << leftMax[i] << ",";

}

// Similarly For getting the right max side for every bars

cout << "again the right valley max values for each bar is - " << endl;

for (int i = n - 2; i >= 0; i--)

{

    rightMax[i] = max(rightMax[i + 1], heights[i + 1]);

    cout << rightMax[i] << ",";

}

cout << rightMax[n - 1] << ",";

cout << endl;

int WaterTrappedd = 0;

for (int i = 0; i < n; i++)

{

```

```

        int currWater = min(leftMax[i], rightMax[i]) - heights[i];

        if (currWater > 0)

        {

            WaterTrappedd += currWater;

        }

    }

    cout << "So, the Water Trapp is - " << WaterTrappedd << endl;

    return WaterTrappedd;

}

```

```

int main()

{

    int n;

    cout << "value of array size" << endl;

    cin >> n;




    int heights[n];

    cout << "Write down the array elemenets - " << endl;

    for (int i = 0; i < n; i++)

    {

        cin >> heights[i];

    }

    cout << "So, the inserted array is - " << endl;

    for (int i = 0; i < n; i++)

    {

        cout << heights[i] << " ";
    }
}

```

```
}

cout << endl;

Trap(heights, n);
```

```
/*
value of array size
```

7

Write down the array elemenets -

4 2 0 6 3 2 5

So, the inserted array is -

4 2 0 6 3 2 5

So, the Left Valley max values for each bar is -

4,4,4,4,6,6,6,again the right valley max values for each bar is -

5,5,5,6,6,6,5,

So, the Water Trapp is - 11

value of array size

5

Write down the array elemenets -

5 4 3 2 1

So, the inserted array is -

5 4 3 2 1

So, the Left Valley max values for each bar is -

5,5,5,5,5,again the right valley max values for each bar is -

1,2,3,4,1,

So, the Water Trapp is - 0

T.C -  $O(n)$

\*/

}

// \_\_\_\_\_

## **Sorting Algorithms -**

- 1) Bubble Sort ALgo -
- 2) Selection Sort Implementation -
- 3) Insertion Sort Algorithm & Inbuilt Sorting technique -
- 4) Counting Sort

### 3 Sorting Algorithms –

#### // 1) Bubble Sort Algo -

The screenshot shows a video player window for '01. Bubble Sort.mp4' in VLC media player. The video frame displays a chalkboard with the title 'Bubble Sort' and a list of numbers [5, 4, 1, 3, 2]. A handwritten note says '(n-1) turns 4'. Below the list, the sequence is shown as [4, 1, 3, 2, 5] with the last element circled. A circular inset in the bottom right corner shows a person in a black shirt speaking into a microphone. The VLC interface at the bottom includes a progress bar at 70% and a 125% zoom level.

The screenshot shows a video player window for '01. Bubble Sort.mp4' in VLC media player. The video frame displays a chalkboard illustrating the steps of a bubble sort algorithm. It shows three stages: 1st turn ([5, 4, 1, 3, 2]), 2nd turn ([4, 1, 3, 2, 5]), and 3rd turn ([1, 3, 2, 4, 5]). The number of turns is noted as '(n-1) turns 4'. A handwritten note indicates the 4th turn with the sequence [1, 2, 3, 4, 5] and a checkmark. A circular inset in the bottom right corner shows a person in a black shirt speaking into a microphone. The VLC interface at the bottom includes a progress bar at 76% and a 125% zoom level.

01. Bubble Sort.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Bubble Sort

(n-1) turns 4

Idea : large elements come to end by swapping with adjacents.

[5, 4, 1, 3, 2]

```
for(int i=0; i<n-1; i++) {  
    for(int j=0; j<n-i-1; j++) {  
        if(a[i] > a[i+1])  
            swap(a[i], a[i+1]);  
    }  
}
```

adj [5, 4, 1, 3, 2]  
1st(i=0) 0 to n-2 <n-1  
2nd(i=1) 0 to n-3 <n-2  
3rd(i=2) 0 to n-4 <n-3  
4th(i=3) 0 to n-5 <n-2

APNA COLLEGE

01. Bubble Sort.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Bubble Sort

(n-1) turns 4

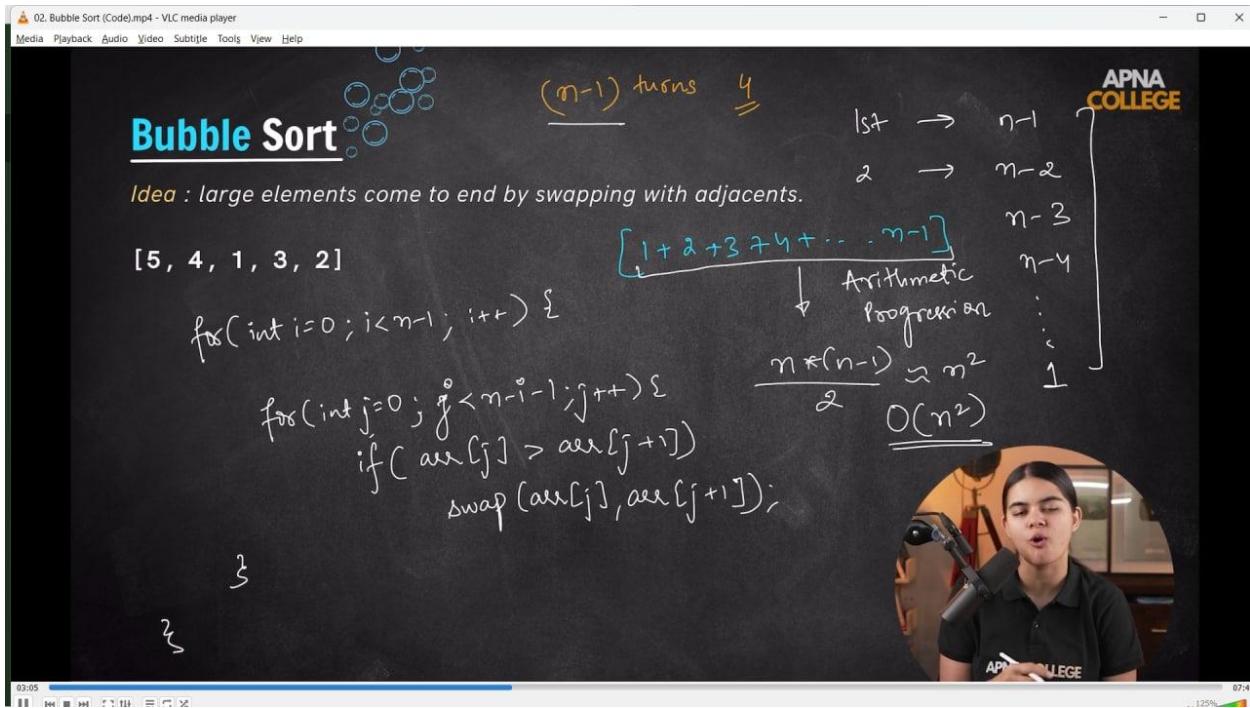
Idea : large elements come to end by swapping with adjacents.

[5, 4, 1, 3, 2]

```
for(int i=0; i<n-1; i++) {  
    for(int j=0; j<n-i-1; j++) {  
        if(a[j] > a[j+1])  
            swap(a[j], a[j+1]);  
    }  
}
```

adj [5, 4, 1, 3, 2]  
1st(i=0) 0 to n-2 <n-1  
2nd(i=1) 0 to n-3 <n-2  
3rd(i=2) 0 to n-4 <n-3  
4th(i=3) 0 to n-5 <n-4

APNA COLLEGE



```
void printArray(int arr[], int n)
```

```
{
```

```
    cout << "The sorted array after operation is - " << endl;
```

```
    for (int i = 0; i < n; i++)
```

```
{
```

```
    cout << arr[i] << " ";
```

```
}
```

```
    cout << endl;
```

```
}
```

```
void bubbleSort(int arr[], int n)
```

```
{
```

```
    for (int i = 0; i < n - 1; i++)
```

```
{
```

```
        bool isSwap = false; // for the cases where array is already sorted, so for avoiding TC issues
```

```
        for (int j = 0; j < n - i - 1; j++)
```

```
{  
    if (arr[j] > arr[j + 1])  
    {  
        swap(arr[j], arr[j + 1]);  
        isSwap = true;  
    }  
}  
  
if (!isSwap)  
{  
    // means array is already sorted, so return it from here  
    return;  
}  
}  
  
printArray(arr, n);  
}
```

```
int main()  
{  
    int n;  
    cout << "value of array size" << endl;  
    cin >> n;  
  
    int arr[n];  
    cout << "Write down the array elements - " << endl;  
    for (int i = 0; i < n; i++)  
    {
```

```
    cin >> arr[i];  
}  
  
cout << "So, the inserted array is - " << endl;  
  
for (int i = 0; i < n; i++)  
{  
    cout << arr[i] << " ";  
}  
  
cout << endl;  
  
bubbleSort(arr, n);
```

return 0;

/\*

value of array size

8

Write down the array elements -

5 4 3 2 1 9 8 6

So, the inserted array is -

5 4 3 2 1 9 8 6

The sorted array is -

1 2 3 4 5 6 8 9

T.C - O( $n^2$ )

\*/

}

// \_\_\_\_\_

```
// For descending Order -  
  
// void printArray(int arr[], int n)  
// {  
//   cout << "The sorted array after operation is - " << endl;  
//   for (int i = 0; i < n; i++)  
//   {  
//     cout << arr[i] << " ";  
//   }  
//   cout << endl;  
// }  
  
// void bubbleSort(int arr[], int n)  
// {  
//   for (int i = 0; i < n - 1; i++)  
//   {  
//     bool isSwap = false; // for the cases where array is already sorted, so for avoiding TC  
// issues  
//     for (int j = 0; j < n - i - 1; j++)  
//     {  
//       if (arr[j] < arr[j + 1])  
//       {  
//         swap(arr[j], arr[j + 1]);  
//         isSwap = true;  
//       }  
//     }  
//     if (!isSwap)
```

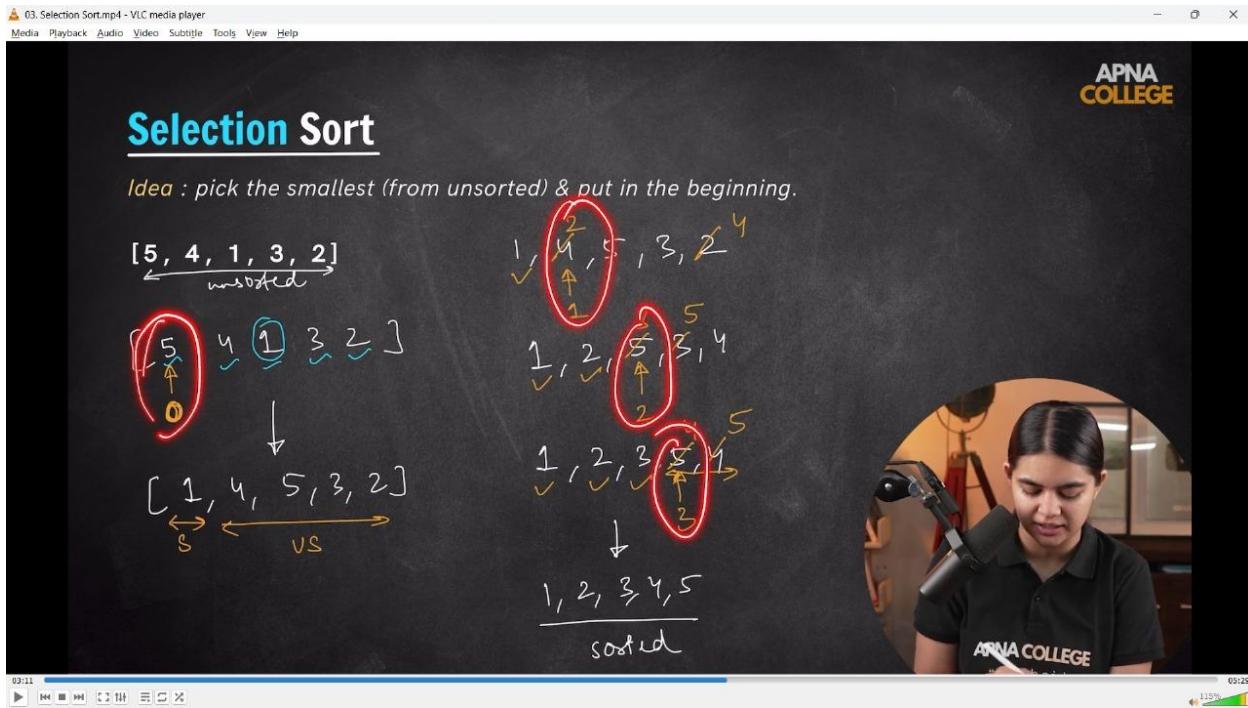
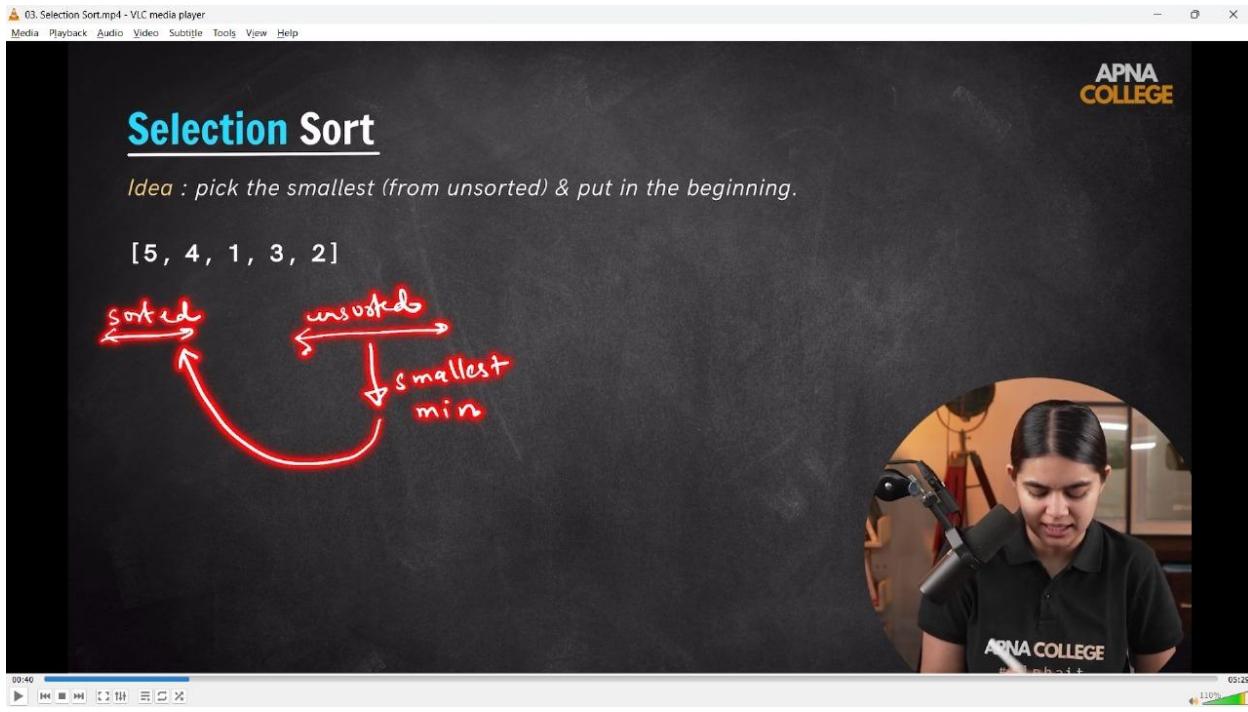
```
//      {
//          // means array is already sorted, so return it from here
//          return;
//      }
//
//  printArray(arr, n);
//
// }

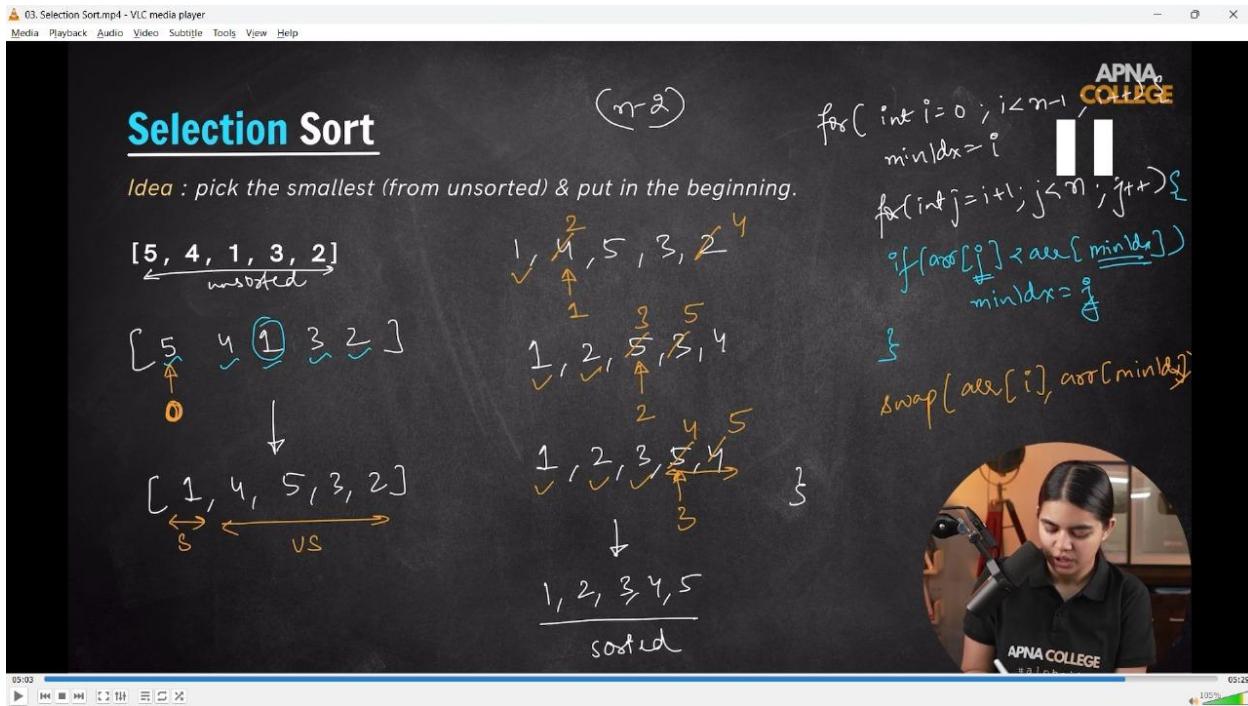
// int main()
//{
//    int n;
//
//    cout << "value of array size" << endl;
//    cin >> n;

//
//    int arr[n];
//
//    cout << "Write down the array elemenets - " << endl;
//
//    for (int i = 0; i < n; i++)
//
//    {
//        cin >> arr[i];
//
//    }
//
//    cout << "So, the inserted array is - " << endl;
//
//    for (int i = 0; i < n; i++)
//
//    {
//        cout << arr[i] << " ";
//
//    }
//
//    cout << endl;
//
//    bubbleSort(arr, n);
```

```
// return 0;  
// /*  
// value of array size  
// 5  
// Write down the array elemenets -  
// 9 5 3 1 4  
// So, the inserted array is -  
// 9 5 3 1 4  
// The sorted array is -  
// 9 5 4 3 1  
  
// */  
// }  
// _____
```

**//2) Selection Sort Implementation –**





```
// void printArray(int arr[], int n)

// {
//   cout << "Array after sorting is - " << endl;
//   for (int i = 0; i < n; i++)
//   {
//     cout << arr[i] << " ";
//   }
//   cout << endl;
// }

// void selectionSort(int arr[], int n)

// {
//   for (int i = 0; i < n - 1; i++)
//   {
//     int minIndex = i;
//     for (int j = i; j < n; j++)
//     {
//       if (arr[j] < arr[minIndex])
//         minIndex = j;
//     }
//     swap(arr[i], arr[minIndex]);
//   }
// }
```

```
//      {
//          if (arr[j] < arr[minIndex])
//          {
//              minIndex = j;
//          }
//      }
//      swap(arr[i], arr[minIndex]);
//  }
//  printArray(arr, n);
// }

// int main()
//{
//    int n;
//    cout << "value of array size" << endl;
//    cin >> n;

//    int arr[n];
//    cout << "Write down the array elements - " << endl;
//    for (int i = 0; i < n; i++)
//    {
//        cin >> arr[i];
//    }
//    cout << "So, the inserted array is - " << endl;
//    for (int i = 0; i < n; i++)
//    {
//        cout << arr[i] << " ";
//    }
//}
```

```
// }  
// cout << endl;  
// selectionSort(arr, n);  
// return 0;  
/*
```

value of array size

5

Write down the array elements -

5 4 1 3 2

So, the inserted array is -

5 4 1 3 2

Array after sorting is -

1 2 3 4 5

\*/

// }

// \_\_\_\_\_

//2.1) Selection Sort Implementation for decreasing Order -

```
void printArray(int arr[], int n)  
{  
    cout << "Array after sorting is - " << endl;  
    for (int i = 0; i < n; i++)  
    {  
        cout << arr[i] << " ";  
    }  
    cout << endl;
```

```
}

void selectionSort(int arr[], int n)

{

    for (int i = 0; i < n - 1; i++)

    {

        int minIndex = i;

        for (int j = i; j < n; j++)

        {

            if (arr[j] > arr[minIndex])

            {

                minIndex = j;

            }

        }

        swap(arr[i], arr[minIndex]);

    }

    printArray(arr, n);

}

int main()

{

    int n;

    cout << "value of array size" << endl;

    cin >> n;

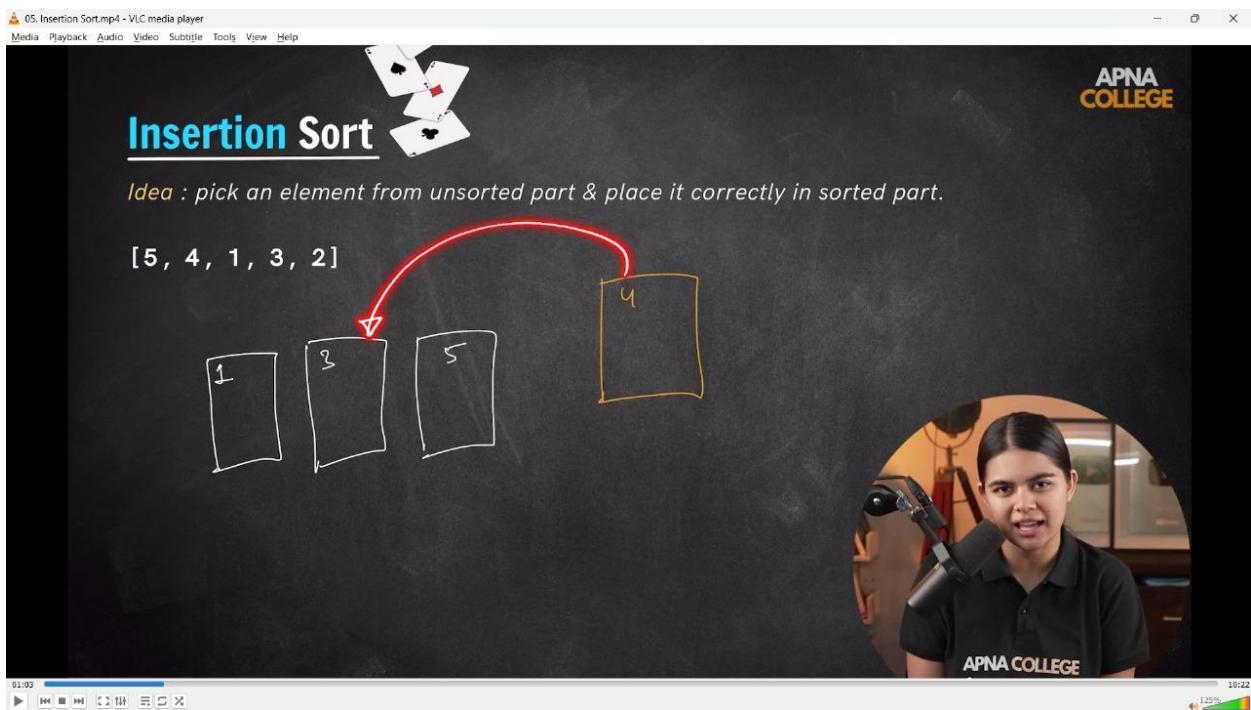



    int arr[n];

    cout << "Write down the array elements - " << endl;

    for (int i = 0; i < n; i++)
```

```
{  
    cin >> arr[i];  
}  
  
cout << "So, the inserted array is - " << endl;  
  
for (int i = 0; i < n; i++)  
{  
    cout << arr[i] << " ";  
}  
  
cout << endl;  
  
selectionSort(arr, n);  
  
return 0;  
  
/*  
value of array size  
5  
  
Write down the array elements -  
1 6 2 4 3 5  
  
So, the inserted array is -  
1 6 2 4 3  
  
Array after sorting is -  
6 4 3 2 1  
  
T.C - O(n*logn) is better than O(n^2).  
*/  
}  
// _____  
  
// 3) Insertion Sort Algorithm -
```



05. Insertion Sort.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Insertion Sort

Idea : pick an element from unsorted part & place it correctly in sorted part.

$[5, 4, 1, 3, 2]$

$\begin{array}{l} \text{S} \\ \cancel{\{5\}}, 1, 3, 2 \end{array}$   $\text{curr} = 4$

$\begin{array}{l} \text{S} \\ \cancel{\{4\}}, 1, 3, 2 \end{array}$   $\text{curr} = 1$

$\begin{array}{l} \text{S} \\ \cancel{\{1\}}, 3, 4, 5, 2 \end{array}$   $\text{curr} = 3$

$\begin{array}{l} \text{S} \\ \cancel{\{1, 2\}}, 3, 4, 5, 2 \end{array}$   $\text{curr} = 2$

$\begin{array}{l} \text{S} \\ \cancel{\{1, 2, 3\}}, 4, 5, 2 \end{array}$   $\text{curr} = 1$

$\begin{array}{l} \text{S} \\ \cancel{\{1, 2, 3, 4\}}, 5, 2 \end{array}$   $\text{curr} = 1$

$\begin{array}{l} \text{S} \\ \cancel{\{1, 2, 3, 4, 5\}} \end{array}$   $\text{curr} = 1$

sorted

APNA COLLEGE

06:37 10:22 125%

05. Insertion Sort.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Insertion Sort

Idea : pick an element from unsorted part & place it correctly in sorted part.

```

[5, 4, 1, 3, 2]

for (int i=1; i<n; i++) {
    curr = arr[i]
    prev = i-1
    while (prev >= 0 && arr[prev] > curr) {
        swap(arr[prev], arr[prev+1])
        prev--;
    }
    arr[curr] = curr;
}

```

APNA COLLEGE

10:20 10:22 125%

```

// void printArray(int arr[], int n)

// {
//     cout << "After Sorting, Array is - " << endl;
//     for (int i = 0; i < n; i++)

```

```
// {
//     cout << arr[i] << " ";
// }
// cout << endl;
//}

// void insertionSort(int arr[], int n)
//{
//     for (int i = 1; i < n; i++)
//     {
//         int curr = arr[i];
//         int prev = i - 1;
//         while (prev >= 0 && arr[prev] > curr)
//         {
//             swap(arr[prev], arr[prev + 1]);
//             prev--;
//         }
//         arr[prev + 1] = curr;
//     }
// }

// printArray(arr, n);
//}

// int main()
//{
//     int n;
//     cout << "value of array size" << endl;
//     cin >> n;
```

```
// int arr[n];
// cout << "Write down the array elements - " << endl;
// for (int i = 0; i < n; i++)
// {
//     cin >> arr[i];
// }
// cout << "So, the inserted array is - " << endl;
// for (int i = 0; i < n; i++)
// {
//     cout << arr[i] << " ";
// }
// cout << endl;
// inseertionSort(arr, n);
```

```
/*
// value of array size
// 5
// Write down the array elements -
// 5 4 1 3 2
// So, the inserted array is -
// 5 4 1 3 2
// After Sorting, Array is -
// 1 2 3 4 5
// */
// }
```

```
// _____
```

```
// 3.1) Insertion Sort Algorithm for descending order -
```

```
// void printArray(int arr[], int n)
// {
//   cout << "After Sorting, Array is - " << endl;
//   for (int i = 0; i < n; i++)
//   {
//     cout << arr[i] << " ";
//   }
//   cout << endl;
// }

// void inseertionSort(int arr[], int n)
// {
//   for (int i = 1; i < n; i++)
//   {
//     int curr = arr[i];
//     int prev = i - 1;
//     while (prev >= 0 && arr[prev] > curr)
//     {
//       swap(arr[prev], arr[prev + 1]);
//       prev--;
//     }
//     arr[prev + 1] = curr;
//   }
// }
```

```
// printArray(arr, n);

// }

// int main()

//{
//    int n;

//    cout << "value of array size" << endl;

//    cin >> n;

//    int arr[n];

//    cout << "Write down the array elemenets - " << endl;

//    for (int i = 0; i < n; i++)

//    {

//        cin >> arr[i];

//    }

//    cout << "So, the inserted array is - " << endl;

//    for (int i = 0; i < n; i++)

//    {

//        cout << arr[i] << " ";

//    }

//    cout << endl;

//    inseertionSort(arr, n);

// /*

// value of array size

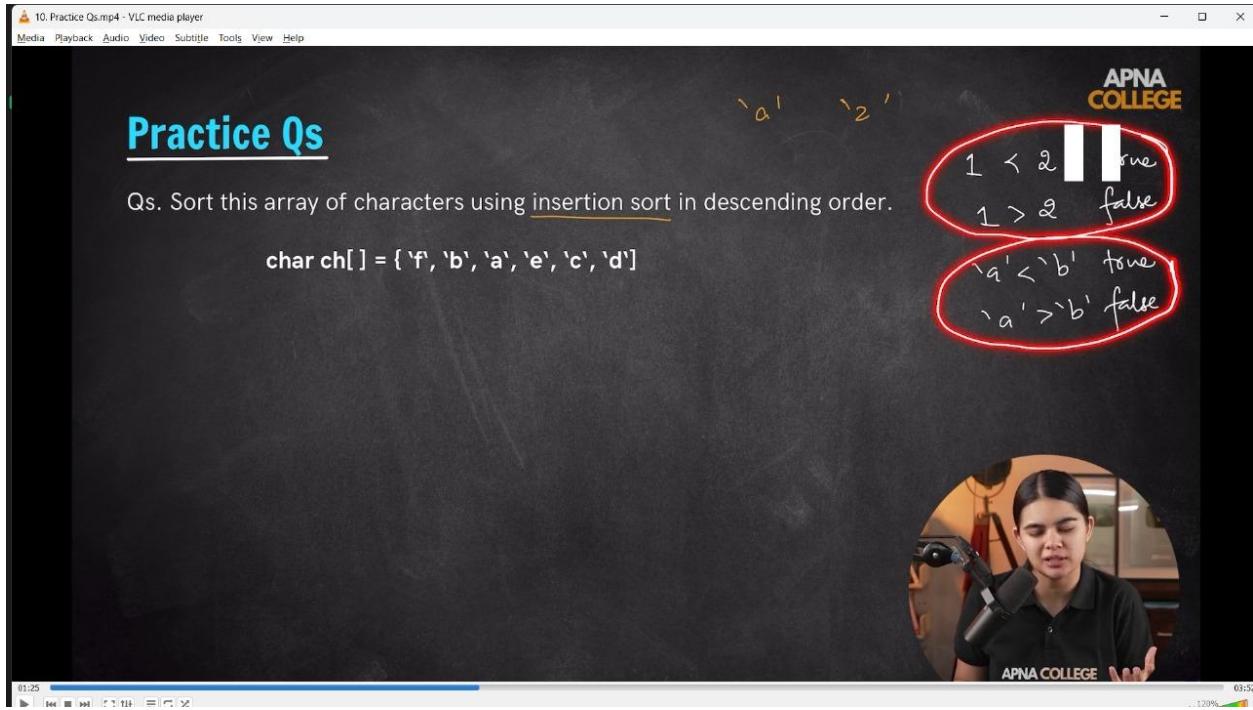
// 5

// Write down the array elemenets -
```

```
// 2 1 4 3 5  
// So, the inserted array is -  
// 2 1 4 3 5  
// After Sorting, Array is -  
// 1 2 3 4 5
```

```
// T.C - O(n^2)  
// */  
// }
```

### **// 3.2) Insertion Sort Practice Qn -**



```
void printArray(char arr[], int n)  
{  
    cout << "After Sorting, Array is - " << endl;  
    for (int i = 0; i < n; i++)
```

```
{  
    cout << arr[i] << ",";  
}  
cout << endl;  
}  
  
void sortChar(char arr[], int n)  
{  
    for (int i = 1; i < n; i++)  
    {  
        int curr = arr[i];  
        int prev = i - 1;  
        while (prev >= 0 && arr[prev] < curr)  
        {  
            swap(arr[prev], arr[prev + 1]);  
            prev--;  
        }  
        arr[prev + 1] = curr;  
    }  
  
    printArray(arr, n);  
}  
  
int main()  
{  
    char ch[6] = {'f', 'b', 'a', 'e', 'c', 'd'};  
    sortChar(ch, 6);  
    return 0;  
}
```

```
/*
```

After Sorting, Array is -

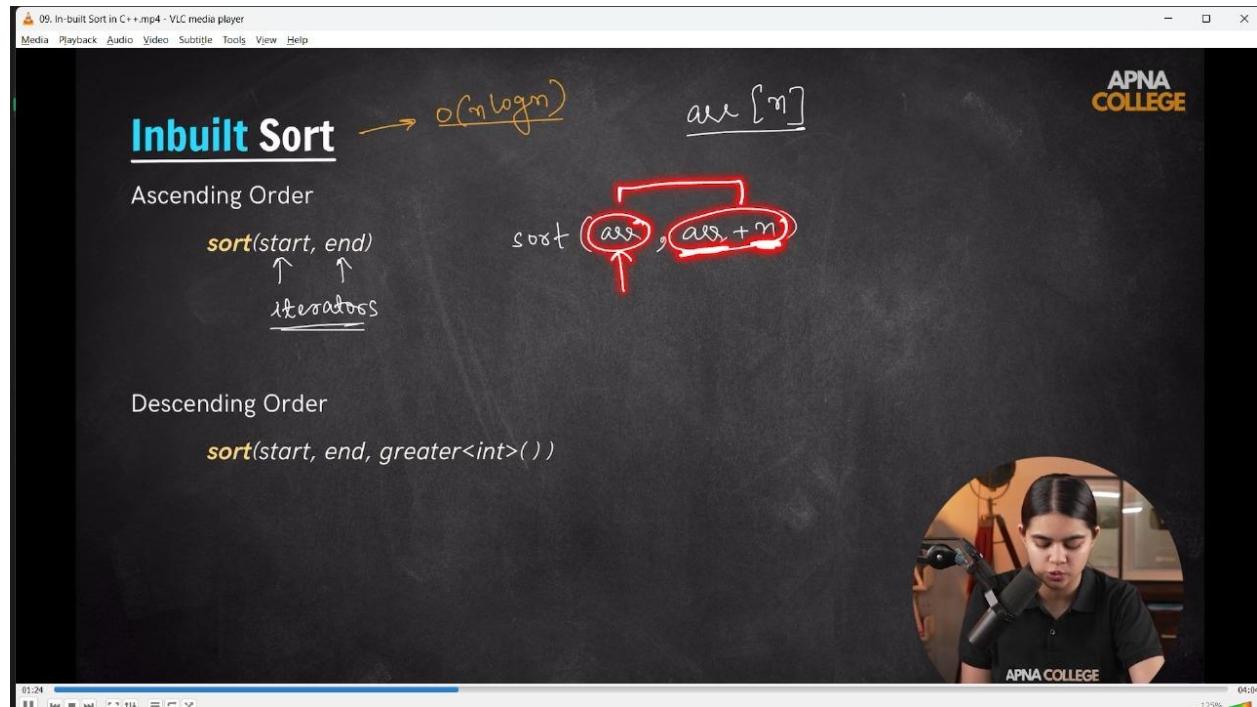
```
f,e,d,c,b,a,
```

```
*/
```

```
}
```

```
// _____
```

#### // 4) in-built sort functionality –



```
// void printArray(int arr[], int n)
```

```
// {
```

```
//   cout << "Array after sorting is - " << endl;
```

```
//   for (int i = 0; i < n; i++)
```

```
//   {
```

```
//     cout << arr[i] << " ";
```

```
//   }
```

```
// cout << endl;  
// }  
// int main()  
// {  
// /* Sort syntax -  
// sort(start, end)  
  
// */  
// int arr[8] = {1, 4, 6, 5, 8, 3, 2, 7};  
// // sort(arr, arr + 8);  
// // printArray(arr,8);  
// /*  
// Array after sorting is -  
// 1 2 3 4 5 6 7 8  
// */  
// sort(arr + 2, arr + 5);  
// printArray(arr, 8);  
// /*  
// Array after sorting is -  
// 1 2 3 4 5 6 7 8  
// */  
  
// // For sorting in decreasing order -  
// // sort(start,end,greater<int>())  
// sort(arr, arr + 8, greater<int>());  
// printArray(arr,8);
```

```
// /*  
// Array after sorting is -  
// 8 7 6 5 4 3 2 1  
// */  
// _____
```

## **Matrix in CPP –**

- 1) 2D Array Basic Introduction & Input, Output
- 2) Spiral matrix -
- 3) Diagonal Sum of Matrix -
- 4) Search for targeted eleemnt in Sorted Matrix -
  - 4.1) - Brute Force Approach - most simple method. T.C -  $O(n*m)$
  - 4.2) - USing Binary Search - 2nd Optimum - Row Wise -  $O(n*\log n)$ , Colm Wise -  $O(m*\log n)$
  - 4.3) - Staircase Search technique -  $O(n+m)$

## Matrix in CPP –

### 1) 2D Array Basic Introduction & Input, Output

The screenshot shows a VLC media player window with a slide titled "2D Arrays". The slide contains the following content:

- A title "2D Arrays" in blue.
- A definition "table / matrix" above two diagrams.
- A "1D" diagram showing a horizontal row of three boxes labeled 1, 2, 3.
- A "2D" diagram showing a 2x3 grid of boxes labeled 1, 2, 3 in the top row and 5, 6, 7 in the bottom row.
- A "3D" diagram showing a 2x2x2 cube with numbers 1 through 8 assigned to its faces.
- A circular video frame in the bottom right corner showing a person speaking.
- Video player controls at the bottom.

The screenshot shows a VLC media player window with a slide titled "2D Arrays". The slide contains the following content:

- A code snippet: 

```
int student[ 2 ][ 3 ] = { { 100, 100, 100 }, { 75, 80, 85 } };
```
- A handwritten note below it: 

```
int student [ 3 ] [ 3 ] = { { 100, 100, 100 }, { 85, 74, 89 }, { 63, 72, 65 } };
```
- A diagram on the right showing the memory layout of a 2D array. It shows a 3x3 grid of values (100, 100, 100, 85, 74, 89, 63, 72, 65) with row and column indices (1, 2, 3). Arrows indicate the row-major traversal of the array.
- A circular video frame in the bottom right corner showing a person speaking.
- Video player controls at the bottom.

01. Introduction to 2D Arrays.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**2D Arrays**

array[4][3]

array[2][1]

APNA COLLEGE

09:29 10:05 125%

APNA COLLEGE #alphait

02. Input & Output a 2D Array.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Input & Output 2D Array**

$n \times m$  [  $3 \times 4$  ]

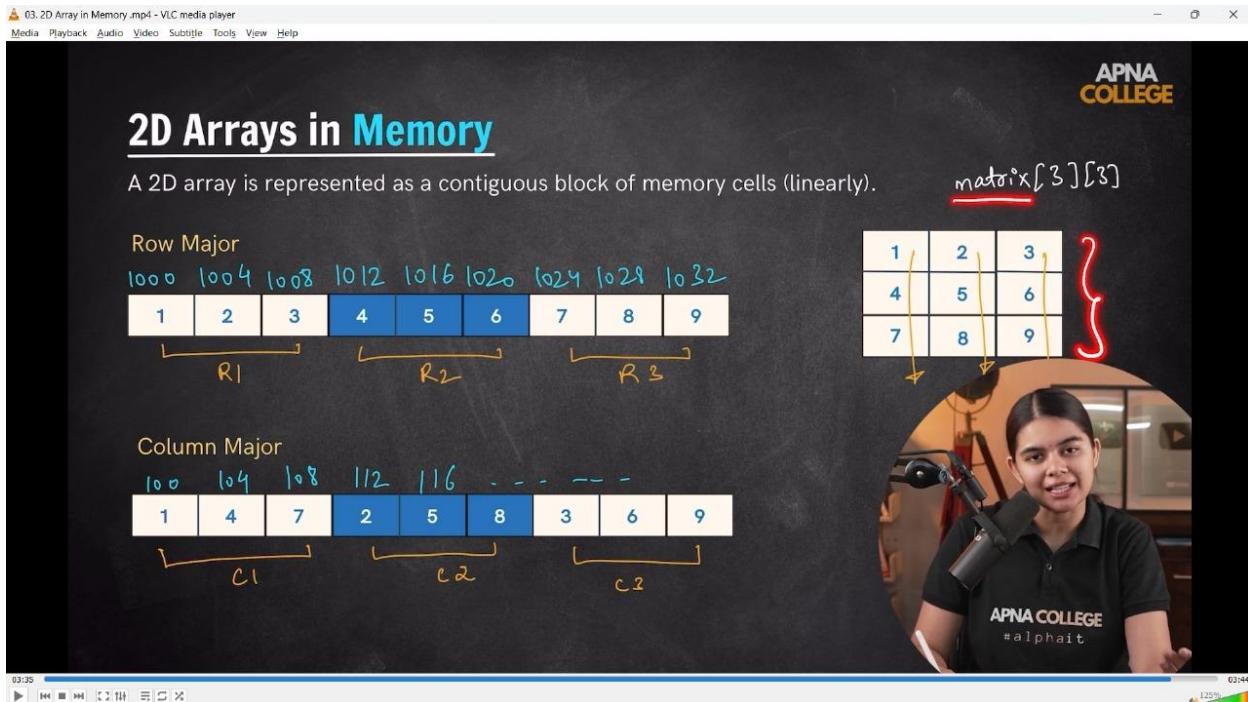
$n \rightarrow \text{rows}$   
 $m \rightarrow \text{columns}$

for (int i = 0 to n-1)  
for (int j = 0 to m-1)

APNA COLLEGE

02:20 05:11 125%

APNA COLLEGE #alphait



```

// int main()
//{
//    int students[3][3] = {{100, 100, 100},
//                          {85, 74, 89},
//                          {63, 72, 65}};
//
//    cout << students[1][2] << endl; // 89
//    cout << students[2][0] << endl; // 63
//
//    // _____
//
//    int n, m;
//
//    cout << "no. of rows & colms are - " << endl;
//    cin >> n >> m;
//
//
//    int arr[n][m];
//
//    // // Taking input & printing the values
//    cout << "Now, enter the array elements - " << endl;

```

```
// for (int i = 0; i < n; i++)
// {
//     for (int j = 0; j < m; j++)
//     {
//         cin >> arr[i][j];
//     }
// }

// cout << "So, the entered 2D Array si - " << endl;

// for (int i = 0; i < n; i++)
// {
//     for (int j = 0; j < m; j++)
//     {
//         cout << arr[i][j] << " ";
//     }
//     cout << endl;
// }

// /*
// no. of rows & colms are -
// 3 3

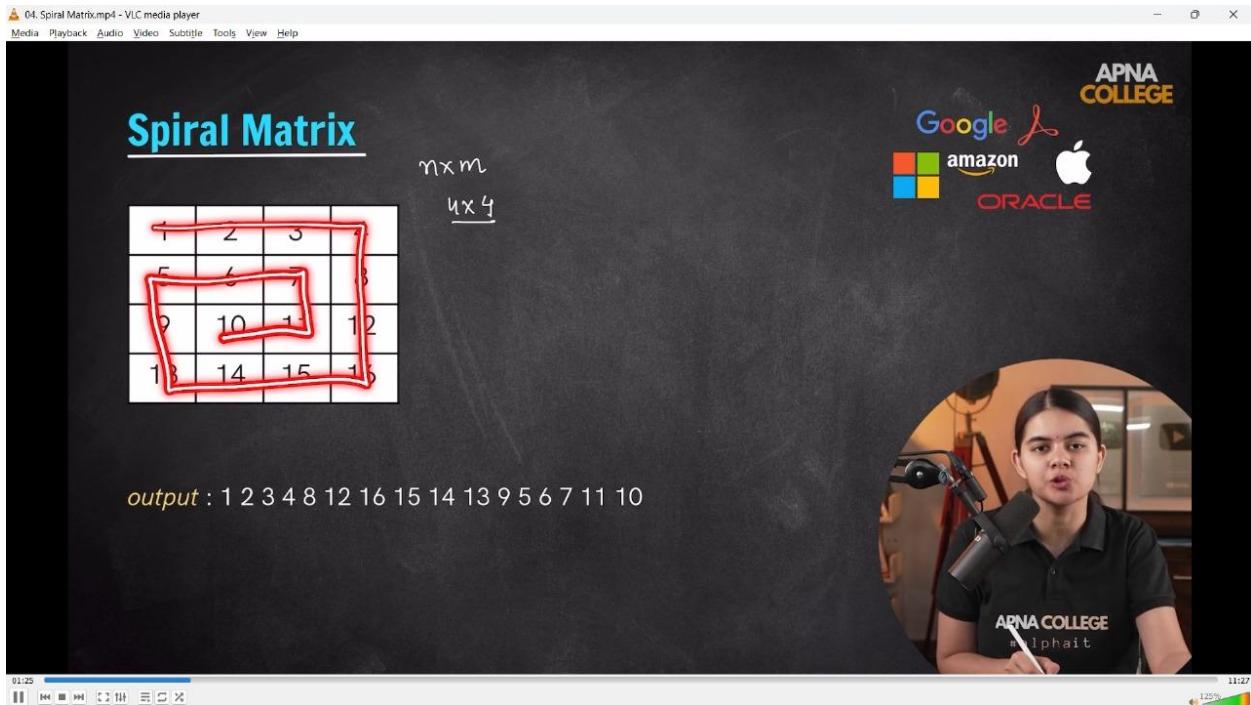
// Now, enter the array elements -
// 1 2 3 4 5 6 7 8 9

// So, the entered 2D Array si -
// 1 2 3
// 4 5 6
// 7 8 9
// */
```

```
// }
```

```
//
```

## 2) Spiral matrix -



04. Spiral Matrix.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Spiral Matrix

Approach

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

APNA COLLEGE

Top ↘  
Right ↙  
Bottom ↖  
Left ↗

APNA COLLEGE  
#alphait

03:14 11:27 125%

04. Spiral Matrix.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Spiral Matrix

Approach

Top  
for ( $s_{row}$  to  $e_{row}$ )  
 $mat[s_{row}][j]$

Right  
for ( $s_{row}+1$  to  $e_{row}$ )  
 $mat[i][e_{col}]$

Bottom  
for ( $e_{col}-1$  to  $s_{col}$ )  
 $mat[e_{row}][j]$

$s_{row}$   $e_{row}$   $s_{row}+1$   $e_{row}+1$   $e_{row}$   $s_{row}+1$   $e_{row}$   $e_{row}+1$

$s_{col}$   $e_{col}$   $s_{col}+1$   $e_{col}+1$   $e_{col}$   $s_{col}+1$   $e_{col}$   $e_{col}+1$

$i$   $j$

$s_{row} = 0$   
 $e_{row} = 0$   
 $e_{row} = n-1 = 3$   
 $e_{col} = m-1 = 3$

APNA COLLEGE  
#alphait

09:30 11:27 125%

04. Spiral Matrix.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Spiral Matrix

Approach

Top  
 $\text{for } (\text{srow to ecol})$   
 $\text{mat}[\text{srow}][\text{j}]$

Right  
 $\text{for } (\text{srow+1 to erow})$   
 $\text{mat}[\text{i}][\text{ecol}]$

Bottom  
 $\text{for } (\text{ecol-1 to scol})$   
 $\text{mat}[\text{erow}][\text{j}]$

Left  
 $\text{for } (\text{erow-1 to srow+1})$   
 $\text{mat}[\text{i}][\text{scol}]$

$srow = 0$   
 $ecol = 0$   
 $erow = n-1 = 3$   
 $ecol = m-1 = 3$

APNA COLLEGE

11:15 11:27 125%

A woman in a black shirt with 'APNA COLLEGE #alphait' is speaking into a microphone.

05. Spiral Matrix (code).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Spiral Matrix

Approach

$srow = 0$   
 $ecol = 3$   
 $erow = 3$   
 $ecol = 0$

$srow = \emptyset$   
 $ecol = 2$   
 $i = 2$   
 $1$

APNA COLLEGE

00:43 17:33 125%

A woman in a black shirt with 'APNA COLLEGE #alphait' is speaking into a microphone.

05. Spiral Matrix (code).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Spiral Matrix

Approach

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

$sRow = 0$     $eRow = 3$   
 $sCol = 0$     $eCol = 3$

APNA COLLEGE

17:33

07:04 07:33

05. Spiral Matrix (code).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Spiral Matrix

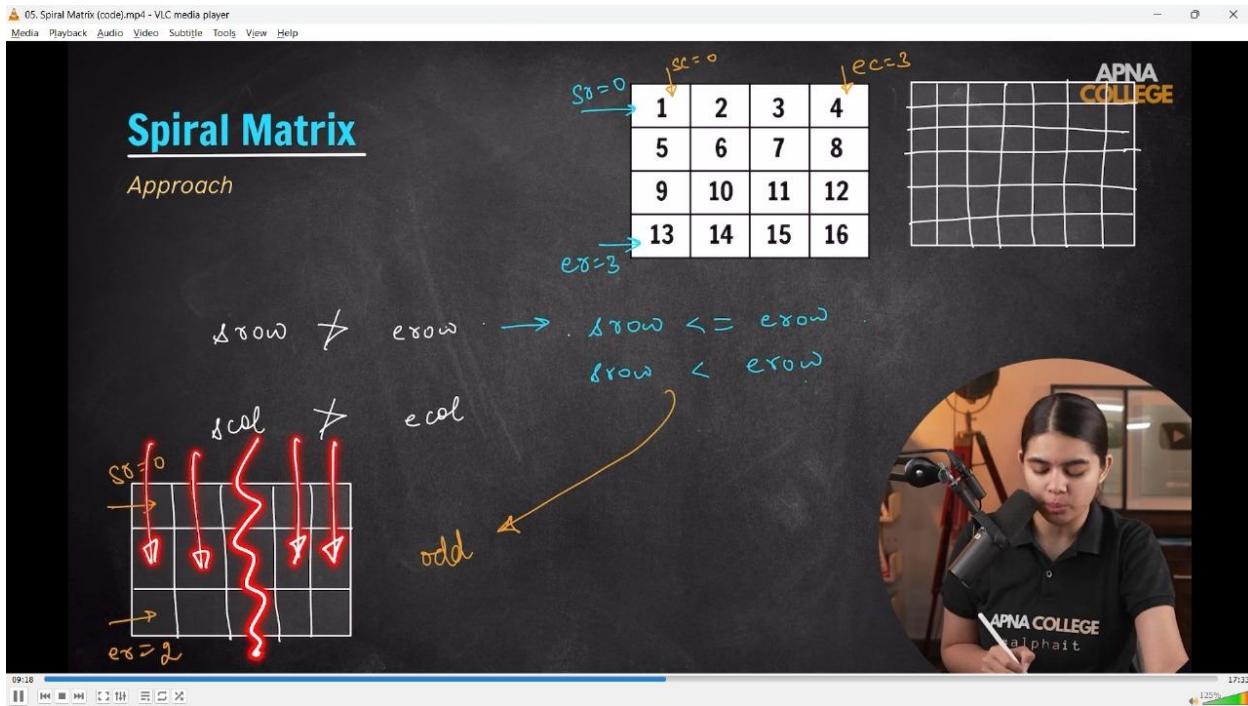
Approach

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

$sRow = 0$     $eRow = 3$   
 $sCol = 0$     $eCol = 3$

APNA COLLEGE

17:33



```
// void spiralMatrix(int mat[100][100], int n, int m)

// {
//     int srow = 0, scol = 0;
//     int erow = n - 1, ecol = m - 1;

//     while (srow <= erow && scol <= ecol) // Odd Matrix - jo value chhoti he vo decide kregi ki
//         loop kitni baar run hoga. taaki kuchh rh na jaayee

//     {
//         // Top
//         for (int j = scol; j <= ecol; j++)
//         {
//             cout << mat[srow][j] << " ";
//         }
//     }

//     // Right
//     for (int i = srow + 1; i <= erow; i++)
//     {
//         // Bottom
//         for (int j = ecol - 1; j >= scol; j--)
//         {
//             cout << mat[i][j] << " ";
//         }
//     }

//     // Left
//     for (int i = erow - 1; i >= srow; i--)
//     {
//         // Top
//         for (int j = ecol - 1; j >= scol; j--)
//         {
//             cout << mat[i][j] << " ";
//         }
//     }

//     // Bottom
//     for (int i = srow; i <= erow - 1; i++)
//     {
//         // Left
//         for (int j = scol + 1; j <= ecol; j++)
//         {
//             cout << mat[i][j] << " ";
//         }
//     }
// }
```

```

//      {
//          cout << mat[i][ecol] << " ";
//      }
//      // Bottom

//      for (int j = ecol - 1; j >= scol; j--)
//      {
//          if (srow == erow) // Middle waali condition(row). yadi middle line printed already then
//          joneed to print it aggain
//
//          {
//              break;
//          }

//          cout << mat[erow][j] << " ";
//      }

//      // Left

//      for (int i = erow - 1; i > srow; i--)
//      {
//          if (srow == erow) // Middle waali condition(col). yadi middle line printed already then
//          joneed to print it aggain
//
//          {
//              break;
//          }

//          cout << mat[i][scol] << " ";
//      }

```

```
//      srow++, erow--;
//      scol++, ecol--;
//  }
//  cout << endl;
//}

// int main()
//{
//  int n, m;
//  cout << "no. of rows & colms are - " << endl;
//  cin >> n >> m;

//  int mat[100][100];
//  // Taking input & printing the values
//  cout << "Now, enter the array elements - " << endl;
//  for (int i = 0; i < n; i++)
//  {
//    for (int j = 0; j < m; j++)
//    {
//      cin >> mat[i][j];
//    }
//  }
//  cout << "So, the entered 2D Array si - " << endl;
//  for (int i = 0; i < n; i++)
//  {
//    for (int j = 0; j < m; j++)
```

```
// {
//     cout << mat[i][j] << " ";
// }
// cout << endl;
// }

// cout << "Now, the spiral matrix moment for this array is - " << endl;
// spiralMatrix(mat, n, m);

// /*
// no. of rows & colms are -
// 4 4

// Now, enter the array elements -
// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

// So, the entered 2D Array si -
// 1 2 3 4
// 5 6 7 8
// 9 10 11 12
// 13 14 15 16

// Now, the spiral matrix moment for this array is -
// 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

// no. of rows & colms are -
// 5 5

// Now, enter the array elements -
// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

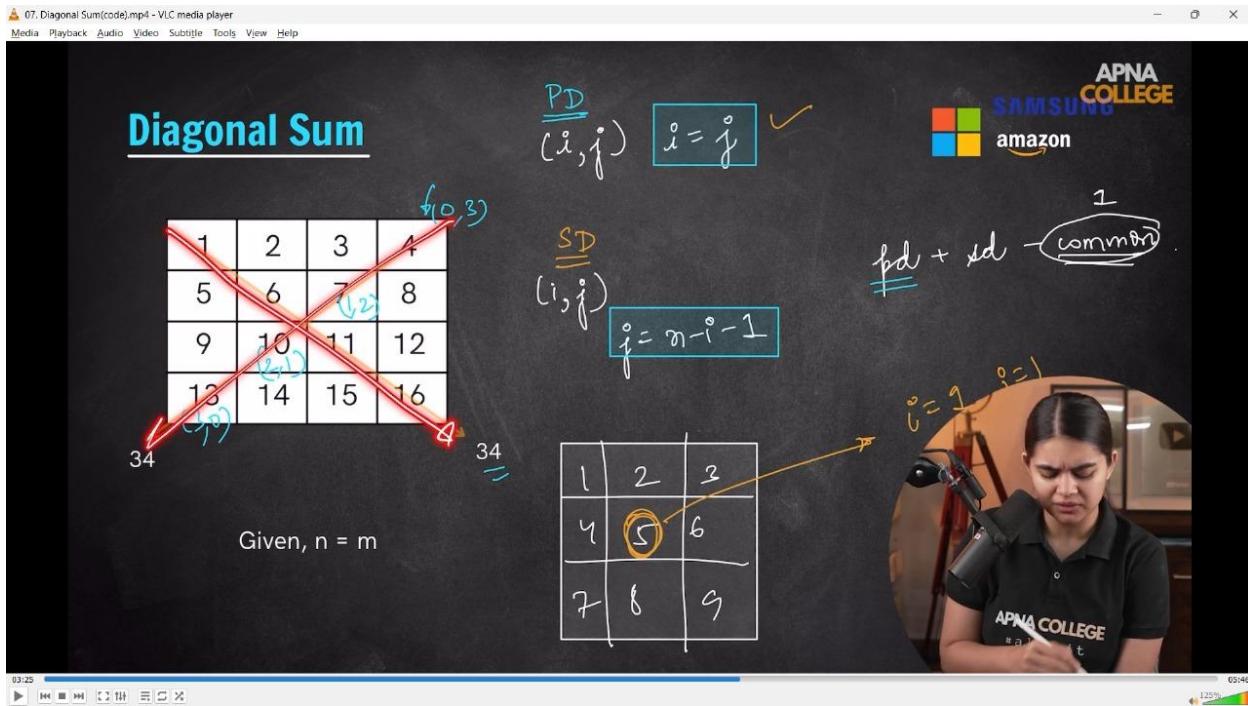
// So, the entered 2D Array si -
// 1 2 3 4 5
```

```
// 6 7 8 9 10  
// 11 12 13 14 15  
// 16 17 18 19 20  
// 21 22 23 24 25  
// Now, the spiral matrix moment for this array is -  
// 1 2 3 4 5 10 15 20 25 24 23 22 21 16 11 6 7 8 9 14 19 18 17 12 13
```

```
// no. of rows & colms are -  
// 3 4  
// Now, enter the array elements -  
// 1 2 3 4 5 6 7 8 9 10 11 12  
// So, the entered 2D Array si -  
// 1 2 3 4  
// 5 6 7 8  
// 9 10 11 12  
// Now, the spiral matrix moment for this array is -  
// 1 2 3 4 8 12 11 10 9 5 6 7
```

```
// */  
//}  
// _____
```

### 3) Diagonal Sum of Matrix -



```
// void diagonalSum(int mat[100][100], int n, int m)

// {
//     int SumofDiagonal1 = 0;
//     int SumofDiagonal2 = 0;
//     int totalSum = 0;
//     int totalDiff = 0;
//     for (int i = 0; i < n; i++)
//     {
//         for (int j = 0; j < m; j++)
//         {
//             if (i == j)
//             {
//                 SumofDiagonal1 += mat[i][j];
//             }
//             if (i + j == n - 1)
//             {
//                 SumofDiagonal2 += mat[i][j];
//             }
//         }
//     }
//     totalSum = SumofDiagonal1 + SumofDiagonal2;
//     totalDiff = Math.abs(SumofDiagonal1 - SumofDiagonal2);
// }
```

```

//      {
//          SumofDiagonal2 += mat[i][j];
//      }
//      }
//      totalSum = SumofDiagonal1 + SumofDiagonal2;
//      totalDiff = SumofDiagonal1 - SumofDiagonal2;
//  }
//  cout << "So, the diagonalSum is - " << totalSum << endl;
//  cout << "and simialrly, the diagonaldiff is - " << totalDiff << endl;

// /* For avoiding complexity from O(n^2)to O(n)
// for(int i=0; i<n; i++)
// {
//     sum+=mat[i][i];
//     if(i!=n-i-1)
//     {
//         sum+=mat[i][n-i-1];
//     }
// }
// */

// }

// int main()
// {
//     int n, m;
//     cout << "no. of rows & colms are - " << endl;

```

```
// cin >> n >> m;

// int mat[100][100];

// // Taking input & printing the values

// cout << "Now, enter the array elements - " << endl;

// for (int i = 0; i < n; i++)

// {

//     for (int j = 0; j < m; j++)

//     {

//         cin >> mat[i][j];

//     }

// }

// cout << "So, the entered 2D Array is - " << endl;

// for (int i = 0; i < n; i++)

// {

//     for (int j = 0; j < m; j++)

//     {

//         cout << mat[i][j] << " ";

//     }

//     cout << endl;

// }

// diagonalSum(mat, n, m);

// }

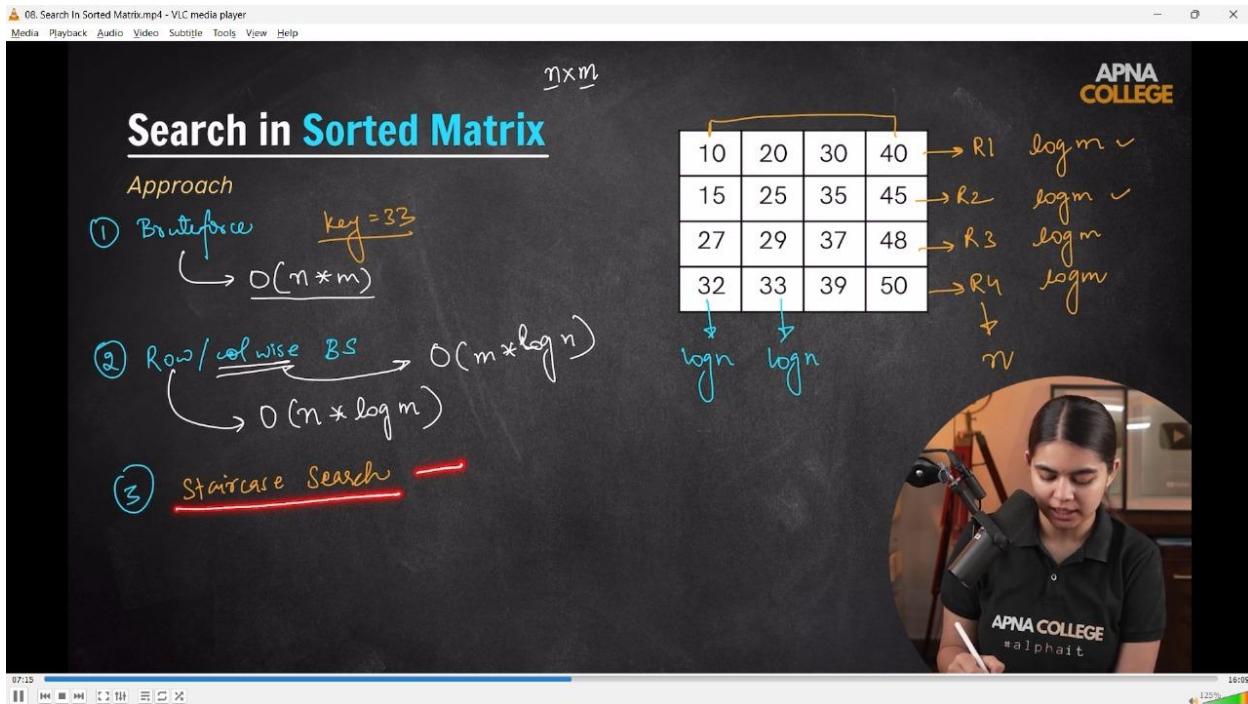
/**/

// no. of rows & cols are -

// 4 4
```

```
// Now, enter the array elements -  
// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
// So, the entered 2D Array si -  
// 1 2 3 4  
// 5 6 7 8  
// 9 10 11 12  
// 13 14 15 16  
// So, the diagonalSum is - 68  
// and similarly, the diagonaldiff is - 0  
  
// no. of rows & colms are -  
// 3 3  
// Now, enter the array elements -  
// 1 2 3 4 5 6 7 8 9  
// So, the entered 2D Array si -  
// 1 2 3  
// 4 5 6  
// 7 8 9  
// So, the diagonalSum is - 30  
// and similarly, the diagonaldiff is - 0  
// T.C - O(n^2)  
// */  
// }  
// _____
```

#### 4) Search for targeted element in Sorted Matrix -



### // 4.1) - Brute Force Approach - most simple method. T.C - O(n\*m)

```

// int targetElementInMatrix(int mat[100][100], int n, int m, int target)

// {

//     // Printing the sum of entered matrix -
//     for (int i = 0; i < n; i++)
//     {
//         for (int j = 0; j < m; j++)
//         {
//             if (mat[i][j] == target)
//             {
//                 cout << "Yes it's present and the address is: row no- " << i << " & column no- " << j
//                 << " having targeted value- " << mat[i][j];
//             }
//         }
//     }
//     return 0;
// }
// }
```

```
// }

// cout << "Sorry, Element Not FOurd Bruh.." << endl;

//}

// int main()

//{
//    int n, m;

//    cout << "no. of rows & colms are - " << endl;

//    cin >> n >> m;

//    int mat[100][100];

//    // Taking input & printing the values

//    cout << "Now, enter the array elements - " << endl;

//    for (int i = 0; i < n; i++)

//    {

//        for (int j = 0; j < m; j++)

//        {

//            cin >> mat[i][j];

//        }

//    }

//    cout << "So, the entered 2D Array si - " << endl;

//    for (int i = 0; i < n; i++)

//    {

//        for (int j = 0; j < m; j++)

//        {

//            cout << mat[i][j] << " ";

//        }

//    }
```

```

//      cout << endl;
//  }

// int target;
// cout << "So, what's the targeted element - " << endl;
// cin >> target;
// targetElementinMatrix(mat,n,m,target);
// /*
// no. of rows & colms are -
// 3 3
// Now, enter the array elements -
// 1 2 3 4 5 6 7 8 9
// So, the entered 2D Array si -
// 1 2 3
// 4 5 6
// 7 8 9
// So, what's the targeted element -
// 6
// Yes it's present and the address is: row no- 1 & column no- 2 having targeted value- 6

// */
// }

// _____
// 4.2) - USing Binary Search - 2nd Optimum - Row Wise - O(n*logn), Colm Wise - O(m*logn)
// rOW major approach for seareching targeted element in the sorted matrix -

```

```
// bool binarySearchInMatrix(int mat[100][100], int n, int m, int target)
//{
//    int start = 0;
//    int end = n * m - 1;

//    while (start <= end)
//    {
//        int mid = start + (end - start) / 2;

//        int row = mid / m;
//        int col = mid % m;

//        if (mat[row][col] == target)
//        {
//            cout << "Yes, it's present at row " << row << " & column " << col << " having targeted
//            value - " << mat[row][col] << endl;
//
//            return true;
//        }
//        else if (mat[row][col] < target)
//        {
//            start = mid + 1;
//        }
//        else
//        {
//            end = mid - 1;
//        }
//    }
//}
```

```
//      }
//  }

//  cout << "Sorry, Element Not Found Bruh.." << endl;
//  return false;
//}

// int main()
//{
//  int n, m;
//  cout << "No. of rows & columns are - " << endl;
//  cin >> n >> m;

//  int mat[100][100];
//  cout << "Now, enter the array elements - " << endl;
//  for (int i = 0; i < n; i++)
//  {
//    for (int j = 0; j < m; j++)
//    {
//      cin >> mat[i][j];
//    }
//  }

//  cout << "So, the entered 2D Array is - " << endl;
//  for (int i = 0; i < n; i++)
//  {
//    for (int j = 0; j < m; j++)
```

```
// {
//     cout << mat[i][j] << " ";
// }
// cout << endl;
// }

// int target;
// cout << "So, what's the targeted element - " << endl;
// cin >> target;

// binarySearchInMatrix(mat, n, m, target);
// /*
// No. of rows & columns are -
// 3 3
// Now, enter the array elements -
// 1 2 3 4 5 6 7 8 9
// So, the entered 2D Array is -
// 1 2 3
// 4 5 6
// 7 8 9
// So, what's the targeted element -
// 8
// Yes, it's present at row 2 & column 1 having targeted value - 8

// */
// }
```

// \_\_\_\_\_

// 4.3) - Staircase Search technique - O(n+m)

// Case 1 - when considering the Top Right Most



08. Search In Sorted Matrix.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Search in Sorted Matrix

Approach

cells special property

cell start  $\Rightarrow \text{mat}[0][m-1]$

cell = key ✓ found

cell < key down row++

cell > key left col--

10	20	30	40
15	25	35	45
27	29	37	48
32	33	39	50

 A yellow box highlights the bottom-right cell (50). To the right of the matrix, the text 'key = 33' is written above two vertical bars."/>

key = 33

APNA COLLEGE

12:46 18:09

125%

08. Search In Sorted Matrix.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Search in Sorted Matrix

Approach

cells special property

cell start  $\Rightarrow \text{mat}[0][m-1]$

while ( $r < n \& c >= 0$ ) {

    cell == key ✓ found

    cell < key down row++

    cell > key left col--

}

10	20	30	40
15	25	35	45
27	29	37	48
32	33	39	50

 A yellow box highlights the bottom-right cell (50). To the right of the matrix, the text 'key = 33' is written above two vertical bars."/>

key = 33

APNA COLLEGE

14:07 18:09

125%

08. Search In Sorted Matrix.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Search in Sorted Matrix

Approach

cells special property

cell start  $\Rightarrow \text{mat}[0][m-1]$

$i \downarrow$        $j \downarrow$

$\text{while } (i < n \text{ & } j >= 0) \{$

$\text{cell} == \text{key}$        $\checkmark \text{ found}$

$\text{cell} < \text{key}$        $\text{down}$   
         $\text{row} + 1$

$\text{cell} > \text{key}$        $\text{left}$   
         $\text{col} - 1$

$\}$

$O(n+m)$

$\text{key} = 33$

$\text{distance} = (n+m)$

10	20	30	40
15	25	35	45
27	29	37	48
32	33	39	50

16:01 16:09

125%

09. Search In Sorted Matrix(Code).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Search in Sorted Matrix

Approach

$n >>> m$   
 $O(n)$

$m >>> n$   
 $= O(m)$

10	20	30	40
15	25	35	45
27	29	37	48
32	33	39	50

06:17 06:18

125%

// Case 1 - when considering the Top Right Most

```
// bool searchStaircase(int mat[100][100], int n, int m, int key)
{
    int i = 0, j = m - 1;
```

```
//  while (i < n && j >= 0)
//
//  {
//    if (mat[i][j] == key)
//
//    {
//      cout << "found at cell (" << i << ", " << j << ")" << endl;
//
//      return true;
//
//    }
//
//    else if (mat[i][j] > key)
//
//    {
//      // as the small value so in the left
//
//      j--;
//
//    }
//
//    else
//
//    {
//      // as greater value so in the right
//
//      i++;
//
//    }
//
//  }
//
//  cout << "key not found" << endl;
//
//  return false;
//
// }
```

```
// int main()
//{
//  int n, m;
//
//  cout << "no. of rows & colms are - " << endl;
```

```
// cin >> n >> m;

// int mat[100][100];

// // Taking input & printing the values

// cout << "Now, enter the array elements - " << endl;

// for (int i = 0; i < n; i++)

// {

//     for (int j = 0; j < m; j++)

//     {

//         cin >> mat[i][j];

//     }

// }

// cout << "So, the entered 2D Array is - " << endl;

// for (int i = 0; i < n; i++)

// {

//     for (int j = 0; j < m; j++)

//     {

//         cout << mat[i][j] << " ";

//     }

//     cout << endl;

// }

// int key;

// cout << "Mention the value you want to search for - " << endl;

// cin >> key;

// searchStaircase(mat, n, m, key);
```

```
/// /*  
// no. of rows & colms are -  
// 4 4  
// Now, enter the array elements -  
// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
// So, the entered 2D Array si -  
// 1 2 3 4  
// 5 6 7 8  
// 9 10 11 12  
// 13 14 15 16  
// Mention. the value you want to search for -  
// 12  
// found at cell (2 , 3)  
// T.C - O(n+m)  
// */  
// }
```

// \_\_\_\_\_

```
// Case 2 - when cosidering the Down Left Corner -  
  
// bool searchStaircase(int mat[100][100], int n, int m, int key)  
// {  
//     int i = n - 1, j = 0;  
//     while (i >= 0 && j < m)  
//     {  
//         if (mat[i][j] == key)
```

```
//      {  
//          cout << "found at cell (" << i << ", " << j << ")" << endl;  
//          return true;  
//      }  
//      else if (mat[i][j] > key)  
//      {  
//          // as the small value so in the left  
//          i--;  
//      }  
//      else  
//      {  
//          // as greater value so in the right  
  
//          j++;  
//      }  
//  }  
//  cout << "key not found" << endl;  
//  return false;  
// }  
  
// int main()  
// {  
//     int n, m;  
//     cout << "no. of rows & colms are - " << endl;  
//     cin >> n >> m;
```

```
// int mat[100][100];

// // Taking input & printing the values

// cout << "Now, enter the array elements - " << endl;

// for (int i = 0; i < n; i++)

// {

//     for (int j = 0; j < m; j++)

//     {

//         cin >> mat[i][j];

//     }

// }

// cout << "So, the entered 2D Array is - " << endl;

// for (int i = 0; i < n; i++)

// {

//     for (int j = 0; j < m; j++)

//     {

//         cout << mat[i][j] << " ";

//     }

//     cout << endl;

// }

// int key;

// cout << "Mention. the value you want to search for - " << endl;

// cin >> key;

// searchStaircase(mat, n, m, key);

// return 0;
```

```
// /*  
// no. of rows & colms are -  
// 4  
// 4  
// Now, enter the array elements -  
// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
// So, the entered 2D Array is -  
// 1 2 3 4  
// 5 6 7 8  
// 9 10 11 12  
// 13 14 15 16  
// Mention. the value you want to search for -  
// 12  
// found at cell (2 , 3)
```

```
// no. of rows & colms are -  
// 4 4  
// Now, enter the array elements -  
// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
// So, the entered 2D Array is -  
// 1 2 3 4  
// 5 6 7 8  
// 9 10 11 12  
// 13 14 15 16  
// Mention. the value you want to search for -  
// 75
```

```

// key not found

// // T.C - O(n+m)

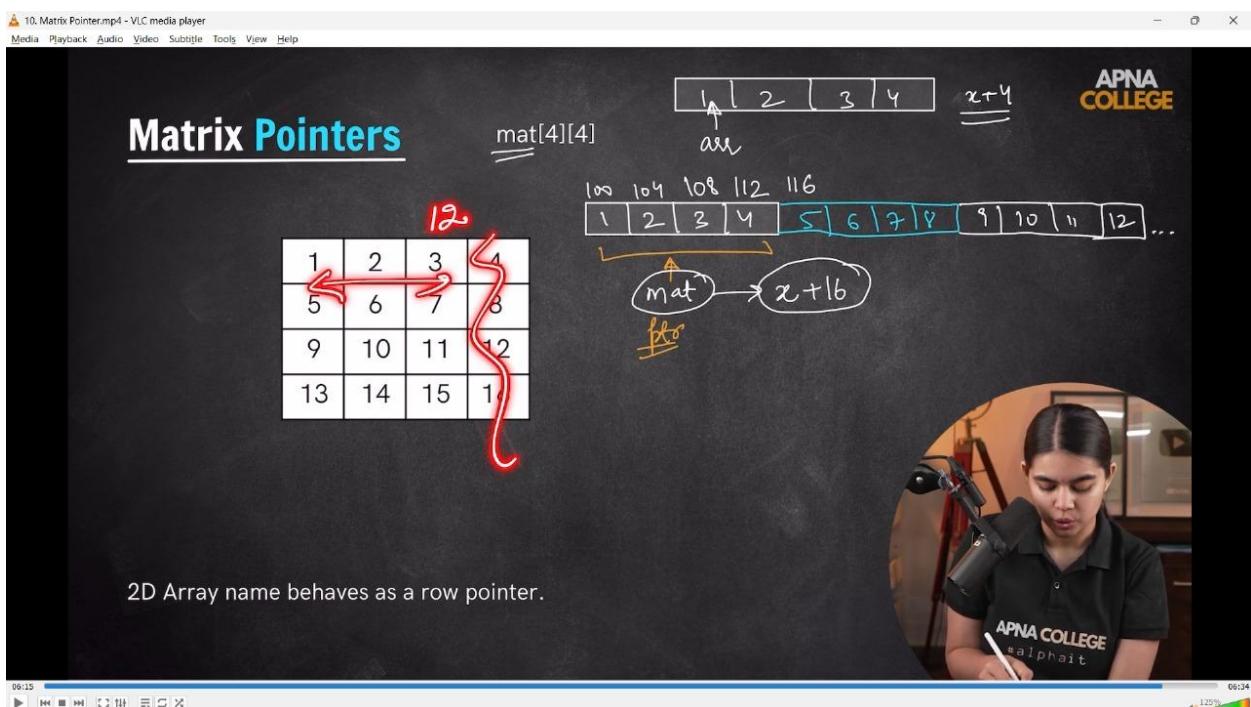
// Special Cases - for TC. When n>>m then TC will be O(n) as row dominating and similarly
when m>>n then TC will be O(m) as cols dominating

// */

// _____

```

### //5) Important pointer concept for matrix –



10. Matrix Pointer.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Cpp Codes

APNA COLLEGE

```

code.cpp x
code.cpp > main()
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int mat[4][4] = {{1, 2, 3, 4},
6                      {5, 6, 7, 8},
7                      {9, 10, 11, 12},
8                      {13, 14, 15, 16}};
9
10    cout << mat << endl;
11    cout << mat+1 << endl;
12    cout << mat+1 << endl;
13    return 0;
14 }
15
16

```

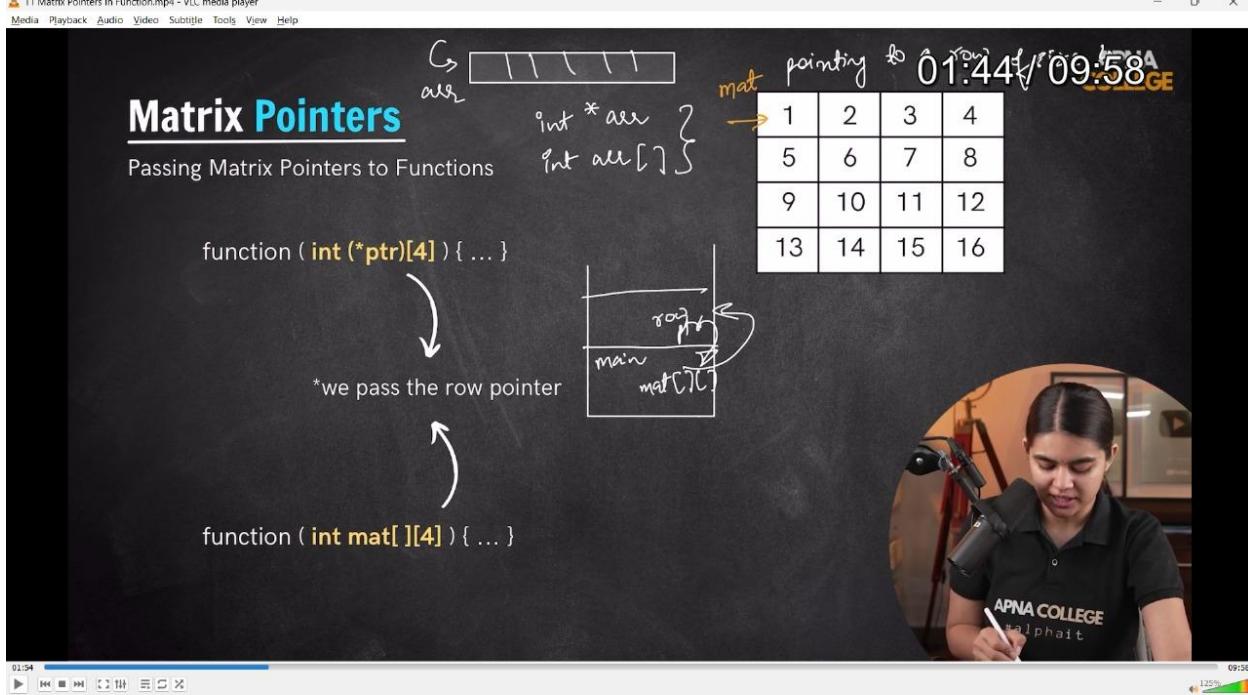
PORTS PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

0x16b7eb318 = 0x16b7eb318
apnacollege@Amans-MacBook-Pro Cpp Codes % g++ code.cpp && ./a.out
0x16f7e7318 = 0x16f7e7318
0x16f7e7328 != 0x16f7e731c
0x16f7e7328 = 0x16f7e7328
apnacollege@Amans-MacBook-Pro Cpp Codes %

```

05:47 06:14 125%



11 Matrix Pointers In Function.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Matrix Pointers

$$\text{ptr}[i][j] = *(*(\text{ptr} + i) + j)$$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

mat → ptr  
 $\underline{(*\text{mat}) \rightarrow \text{row}}$

11 Matrix Pointers In Function.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Matrix Pointers $(i, j)$

$$\text{ptr}[i][j] = *(*(\text{ptr} + i) + j)$$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

mat    ith row value  
 $*(*(\text{mat} + i) + j)$

$*(*(\text{mat} + 2) + 2)$

```
// void func(int mat[][4], int n, int m)
{
    cout << "0th row ptr" << mat << endl; // 0th row ptr 0x61fee0
```

```
// cout << "1st row ptr" << mat + 1 << endl;//1st row ptr0x61fef0
// cout << "2nd row ptr" << mat + 2 << endl;//2nd row ptr0x61ff00
// cout << endl;

// cout << "0th row value is - " << *mat << endl;//0th row value is - 0x61fee0
// cout << "1st row value is - " << *(mat + 1) << endl;//1st row value is - 0x61fef0
// cout << "2nd row value is - " << *(mat + 2) << endl;//2nd row value is - 0x61ff00
// cout << endl;

// cout << *(*(mat + 2) + 2) << endl;//11
// }

// void func2(int (*mat)[4], int n, int m)
// {
// }

// int main()
// {
//     int mat[4][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};
//     /*
//     Above mentioned both the way of passing matrix into
//     user defined function are same. Any of the above can be used
//     */
//     func(mat, 4, 4);

//     /*

```

```
// 0th row ptr0x61fee0  
// 1st row ptr0x61fef0  
// 2nd row ptr0x61ff00
```

```
// */
```

```
// }
```

```
// _____  
_____
```

## **[5] - Character Array & Strings in CPP –**

- 1) Intro to Char Array –
- 2) Covert to Upper Letter -
- 3) Reverse a Char Array -
- 4) Valid Palindrome in char array -
- 5) Char Array(String) Functions -
- 6)String Intro & Basic I/p, o/p operations - 6.1) String Member Functions-
- 7) Check the two strings are Anagram or not –

## 5) Character Array & Strings in CPP –

### // 1) Intro to Char Array –

01.Revisit Char Data Type.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Char Data Type

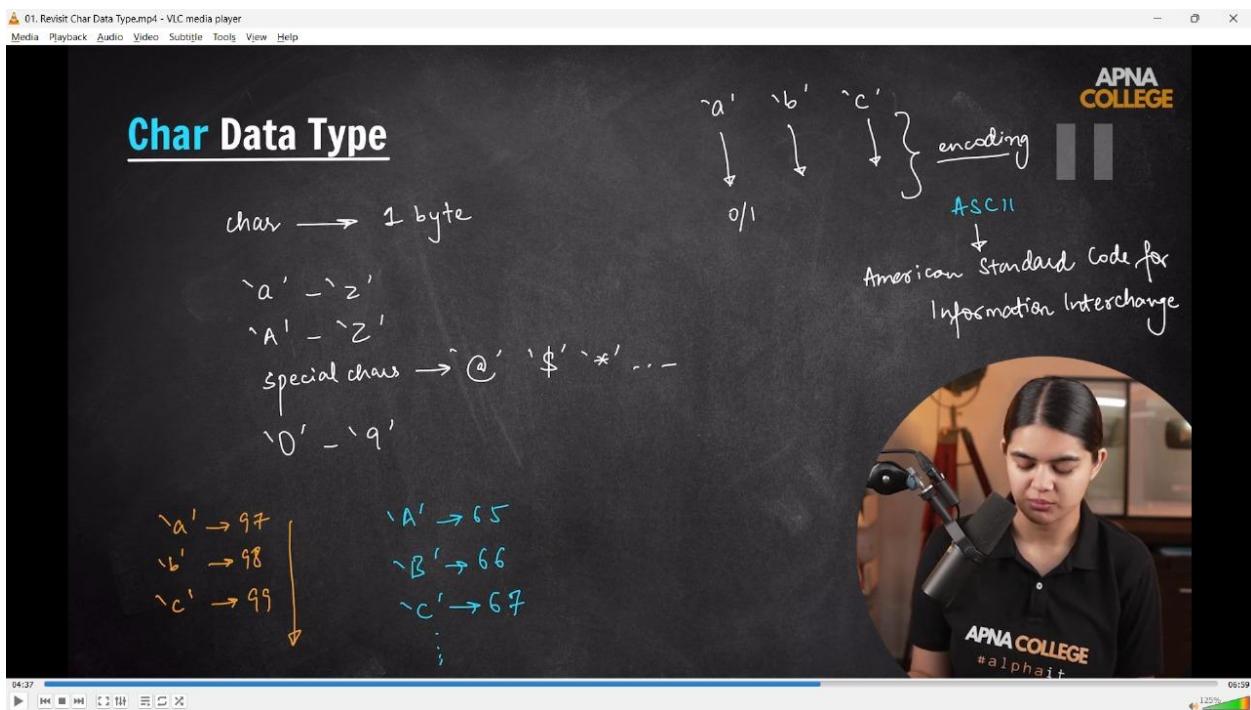
char → 1 byte

'a' - 'z'  
'A' - 'Z'  
special chars → '@' '\$' '\*' '-'  
'0' - '9'

'a' → 97  
'b' → 98  
'c' → 99

'A' → 65  
'B' → 66  
'C' → 67

↓                    ↓  
0/1                 } encoding  
                       ASCII  
                       American Standard Code for  
                       Information Interchange



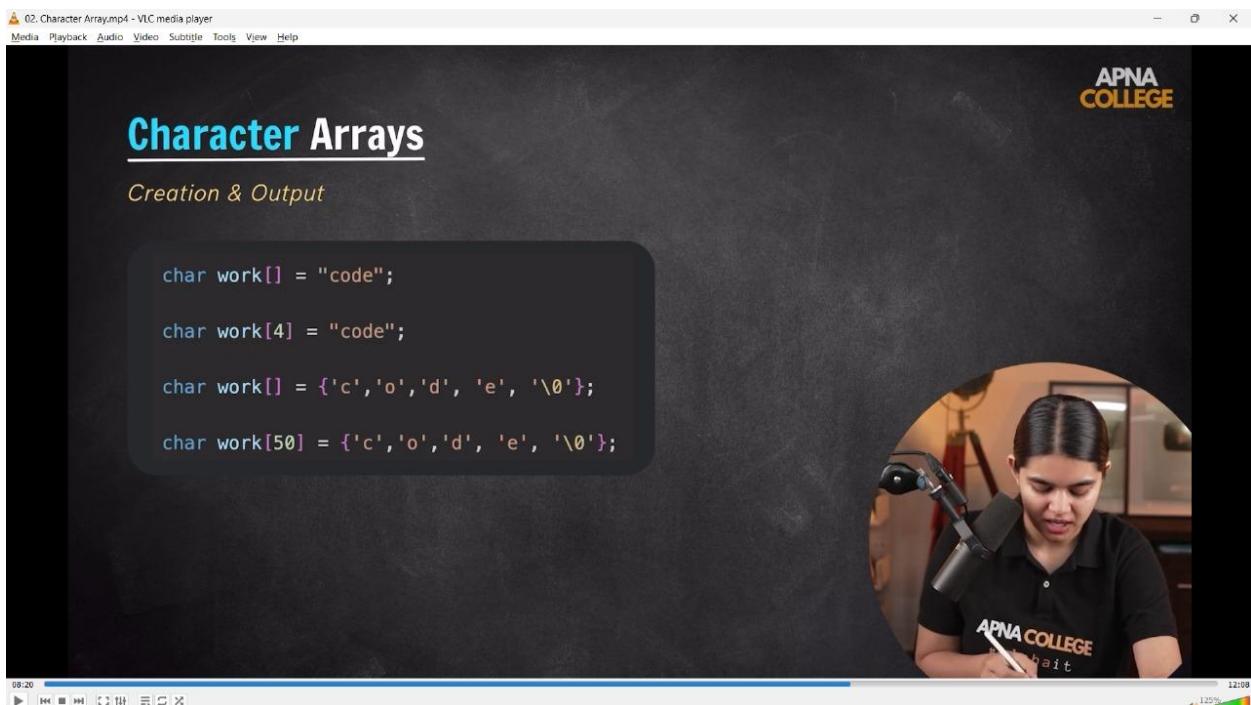
02.Character Array.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Character Arrays

*Creation & Output*

```
char work[] = "code";  
  
char work[4] = "code";  
  
char work[] = {'c', 'o', 'd', 'e', '\0'};  
  
char work[50] = {'c', 'o', 'd', 'e', '\0'};
```





```
// int main()
//{
//    char ch = 'F';
//    int pos = ch - 'A';
//    cout << pos << endl; // 5

//    char arr[5] = {'a', 'b', 'c', 'd', 'e'};
//    cout << arr[0] << endl;
//    cout << arr[1] << endl;
//    cout << arr[2] << endl;
//    cout << arr[3] << endl;
//    cout << arr[4] << endl;
//    cout << arr[5] << endl;

//    char arr2[8] = {'s', 'h', 'u', 'b', 'h', 'a', 'm'};
```

```

// cout << arr2 << endl; // shubham - in int array on printing array, it gives address of array in
hexadecimal form, but in char array it directly gives the value

// // at the end - /0 indicates that these are not letters only this is a valid string word.
// char arr3[] = {'m', 'a', 'h', 'a', 'j', 'a', 'n', '\0'};
// cout << arr3 << endl; // mahajan

// "Microsoft"; // it's a string literal -literal means constant. which is a fix value
// // String creation methods -
// char arru[] = {'M', 'i', 'c', 'r', 'o', 's', 'o', 'f', 't', '\0'};
// char arru2[12] = {'H', 'y', 'd', 'e', 'r', 'a', 'b', 'a', 'd'};
// char work[] = "SoftwareDeveloper";
// char office[] = "New Building";

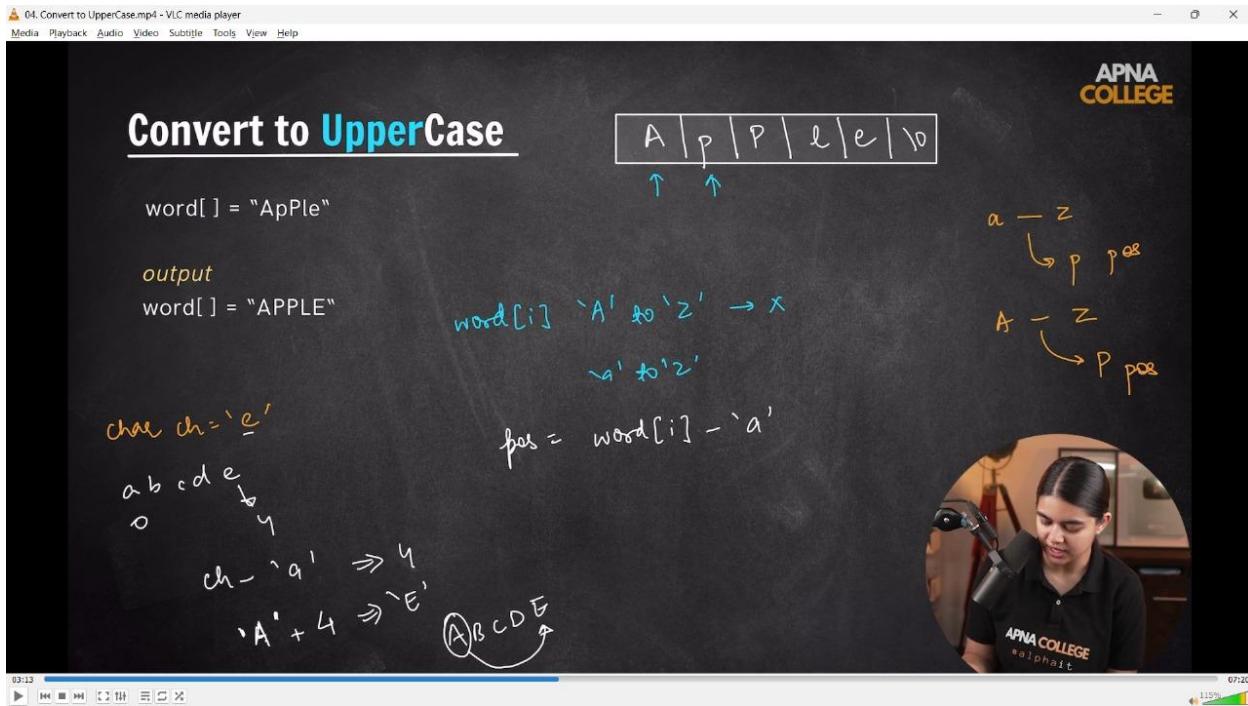
// cout << arru << endl;      // Microsoft
// cout << work << endl;      // SoftwareDeveloper
// cout << arru2 << endl;      // Hyderabad
// cout << office << endl;      // NewBuilding
// cout << strlen(work) << endl; // 17
// cout << strlen(office) << endl; // 12

// // taking input on string -
// /* char shabd[15];
// cin >> shabd;
// cout<<"So, you typed - "<<shabd<<"and the length is - "<<strlen(shabd)<<endl; */

```

```
// /*  
//  Pune  
//  So, you typed - Puneand the length is - 4  
// */  
  
// // for the input of full statement  
// /* char carrier[50];  
// cin.getline(carrier, 50);  
// cout << "So, your entered string is - " << carrier << endl; // in Microsoft with the package of  
51 LPA */  
  
// // If i want to stop meanwhile the sentence  
// char carrier2[50];  
// cin.getline(carrier2, 50, '*');  
// cout << "Entered String is - " << carrier2 << endl;  
// /*  
// Only in Microsoft with * the package of 51 LPA  
// Entered String is - Only in Microsoft with  
// */  
// }  
// _____
```

## 2) Covert to Upper Letter -



```
// void toUpper(char word[], int n)

// {
//     for (int i = 0; i < n; i++)
//     {
//         char ch = word[i];
//         if (ch >= 'A' && ch <= 'Z') // if its already in Uppper Case
//         {
//             continue;
//         }
//         else
//         {
//             word[i] = ch - 'a' + 'A';
//         }
//     }
// }
```

```
// int main()
//{
//    cout << "What's in your mind - " << endl;
//    char word[50];
//    cin.getline(word, 50);

//    toUpper(word, strlen(word));
//    cout << word << endl;
//    /*
//    What's in your mind -
//    MicrosoftHyderabad
//    MICROSFTHYDERABAD

//    */
//}
```

---

**// 2.1) Covert to Lowercase Letetr -**

```
// void toUpper(char word[], int n)
//{
//    for (int i = 0; i < n; i++)
//    {
//        char ch = word[i];
//        if (ch >= 'a' && ch <= 'z') // if its already in Lower Case
//        {
//            continue;
//        }
//    }
//}
```

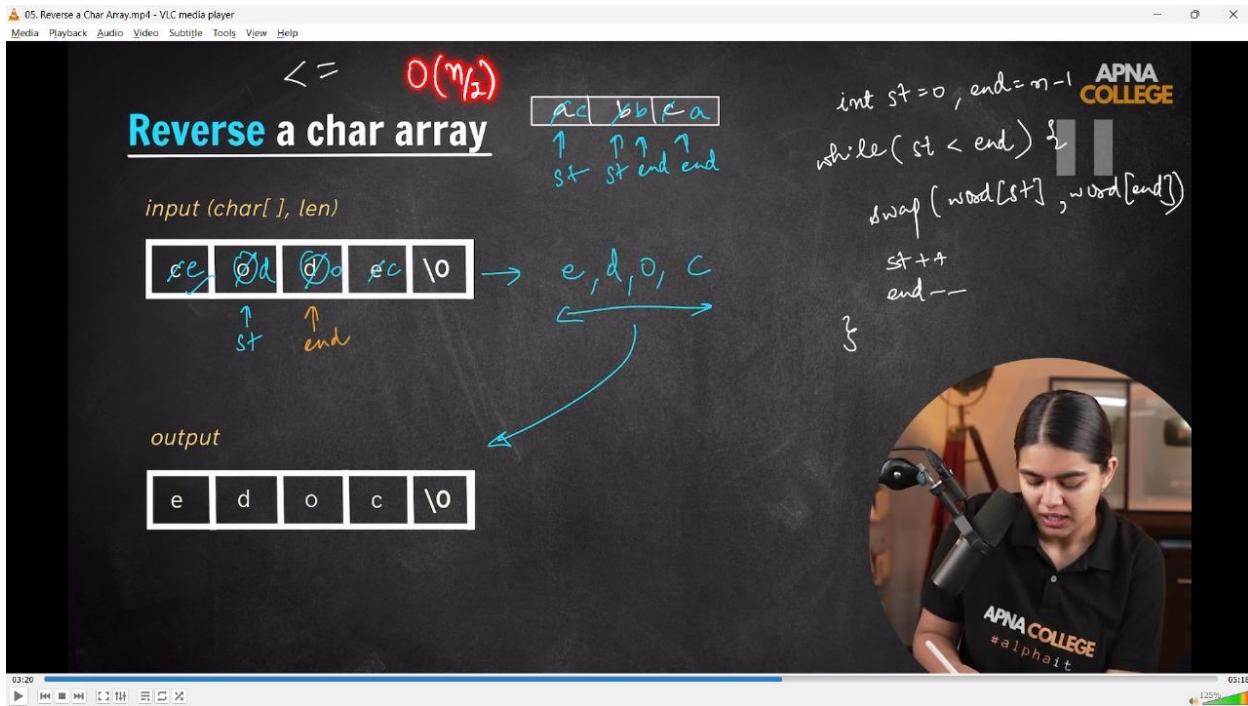
```
//      else
//      {
//          word[i] = ch - 'A' + 'a';
//      }
// }

// int main()
//{
//    cout << "What's in your mind - " << endl;
//    char word[50];
//    cin.getline(word, 50);

//    toUpper(word, strlen(word));
//    cout << word << endl;
//    /*
//    What's in your mind -
//    MicrosoftHyderabad
//    microsfthyderabad

//    */
// }
```

### 3) Reverse a Char Array -



```
// void ReverseCharArray(char word[], int n)

// {
//     int st = 0, end = n - 1;
//     while (st < end)
//     {
//         swap(word[st], word[end]);
//         st++;
//         end--;
//     }
//     cout << "Hence, the reversed word is - " << word << endl;
// }

// int main()
// {
//     cout << "What's in your mind - " << endl;
// }
```

```

//  char word[50];
//  cin.getline(word, 50);

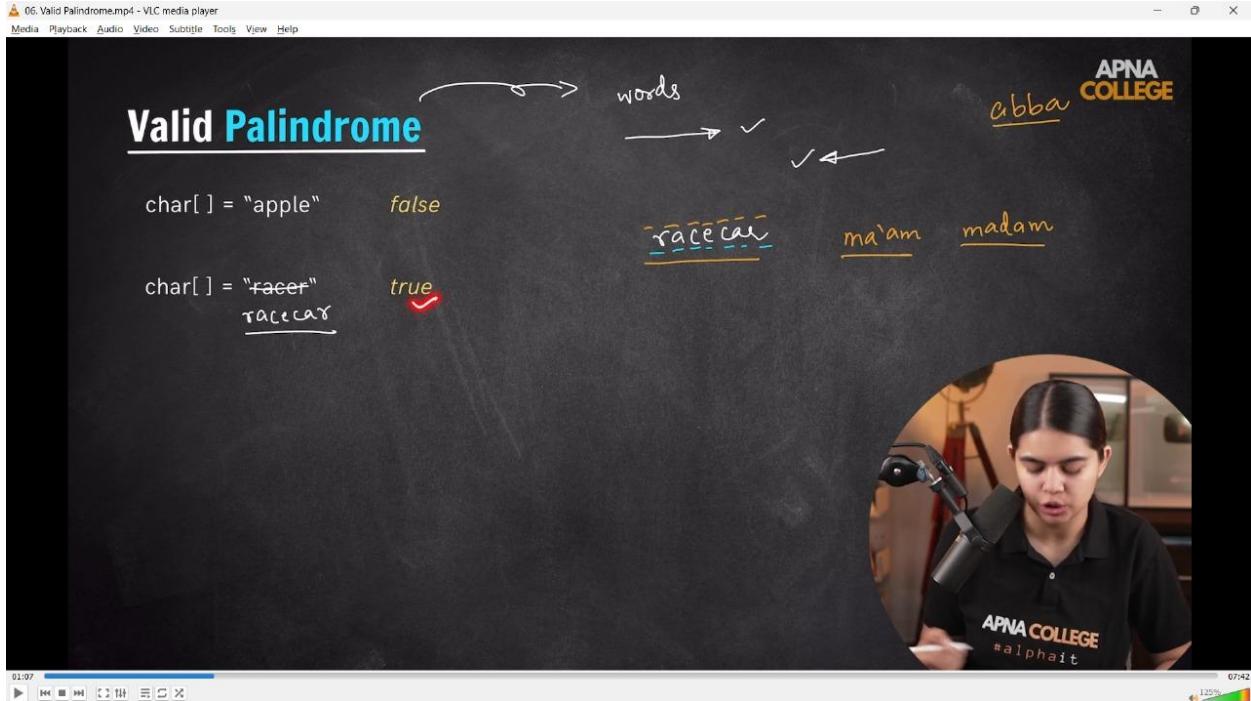
// ReverseCharArray(word, strlen(word));
// /*

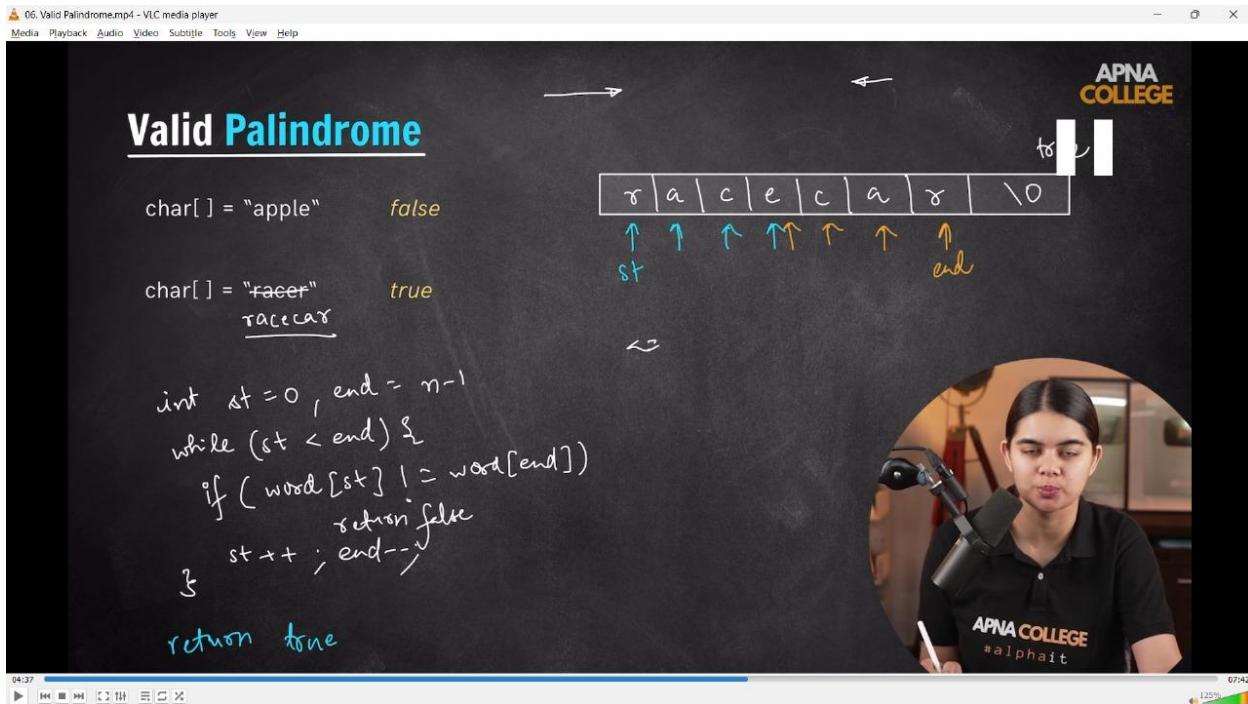
// What's in your mind -
// MicrosoftHyderabad
// Hence, the reversed word is - dabaredyHtfosorciM

// */
// }
// -----

```

#### 4) Valid Palindrome in char array -





```
// bool PalindromeWord(char word[], int n)

// {
//     int st = 0, end = n - 1;
//     while (st < end)
//     {
//         if (word[st] != word[end])
//         {
//             cout << "Not valid Palindrome " << endl;
//             return false;
//         }
//         st++;
//         end--;
//     }
//     cout << "Valid Palindromed " << endl;
//     return true;
}
```

```
// }

// int main()
//{
//    cout << "What's in your mind - " << endl;
//    char word[100];
//    cin.getline(word, 100);
//    PalindromeWord(word, strlen(word));
//    /*
```

```
//    What's in your mind -
//    NaMaN
//    Valid Palindromed
```

```
//    What's in your mind -
//    naman
//    Valid Palindromed
```

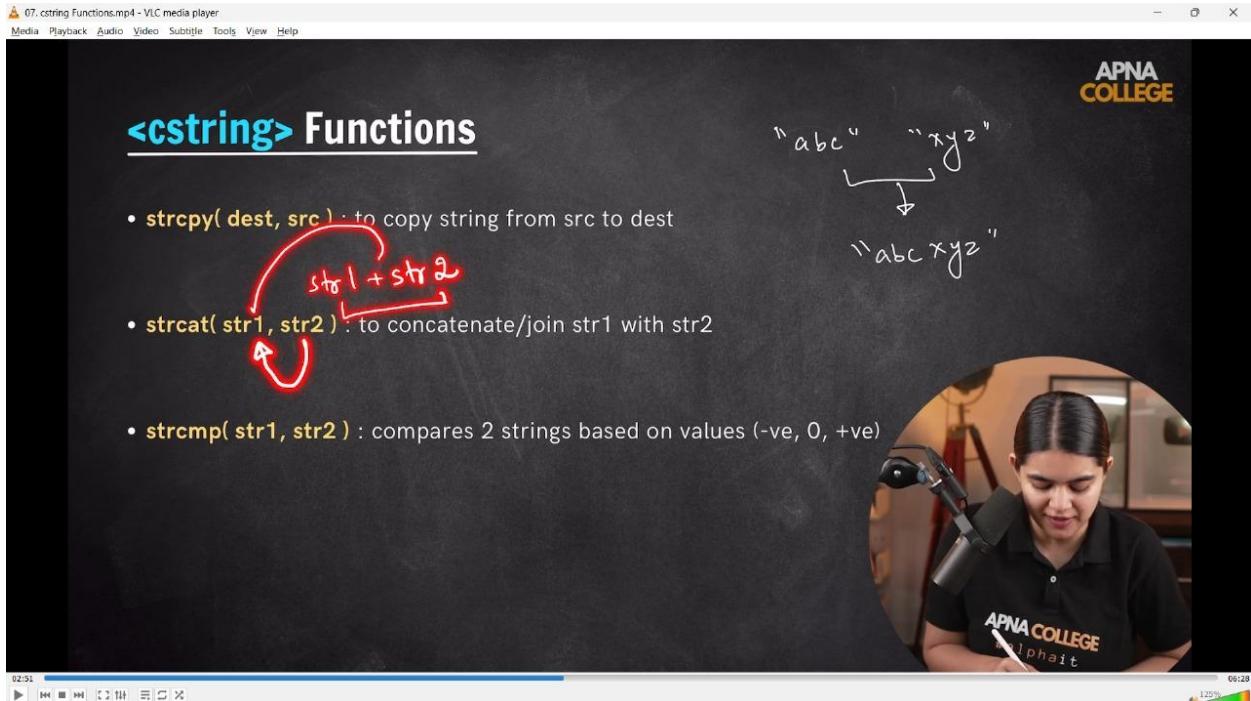
```
//    What's in your mind -
//    Naman
//    Not valid Palindrome
```

```
//    What's in your mind -
//    racecar
//    Valid Palindromed
```

```
//    */
```

```
// }  
//
```

## 5) Char Array(String) Functions -



```
// int main()  
// {  
//     char word1[100];  
//     // str1 = "Shubham Mahajan"; - as it's showing the error in assigning value, so for allowing  
//     it we can use cstring functions - that is strcpy  
//     char word2[100] = "Microsoft Hyderabad";  
//     strcpy(word1, "Free Offcie tour");  
//     cout << word1 << endl; // Free Offcie tour  
  
//     // For String Concatenation -  
//     char word3[100] = "Let's come inside";  
//     strcat(word2, word3);
```

```
// cout << word2 << endl; // Microsoft HyderabadLet's come inside

// // For String Comparision -
// char word4[] = "Microsoft Hyderabad";
// cout << strcmp(word2, word4) << endl; // 1

// char word5[100] = "abc";
// char word6[100] = "xyz";
// cout<<strcmp(word5,word6)<<endl;//-1 - if the baad waali string is big, then returns
negative value

// char word7[100] = "xyz";
// char word8[100] = "abc";
// cout<<strcmp(word7,word8)<<endl;//1 - if the phle waali string is big, then returns
positive value

// }
```

---

## 6)String Intro & Basic I/p, o/p operations –

09. C++ Strings.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## String in C++

C++ Strings are objects of pre-defined string class in **STL**.

C++ Strings have useful **member functions**.

C++ Strings are **dynamic** (their size can change at run time).

C++ Strings support **operators** like +, ==, >, < etc.

C++ Strings are stored **contiguously** in memory.

String str = "hello"; APNA COLLEGE

APNA COLLEGE

01:45 06:13 125%

08. OOPS intro (for Strings).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## String in C++

OOPS : Class, Object, Member Functions & Properties

**blueprint**  $\Rightarrow$  start(), Stop()

- car1 : color : "black"
- car2 : color : "pink"
- car3 : color : "white"

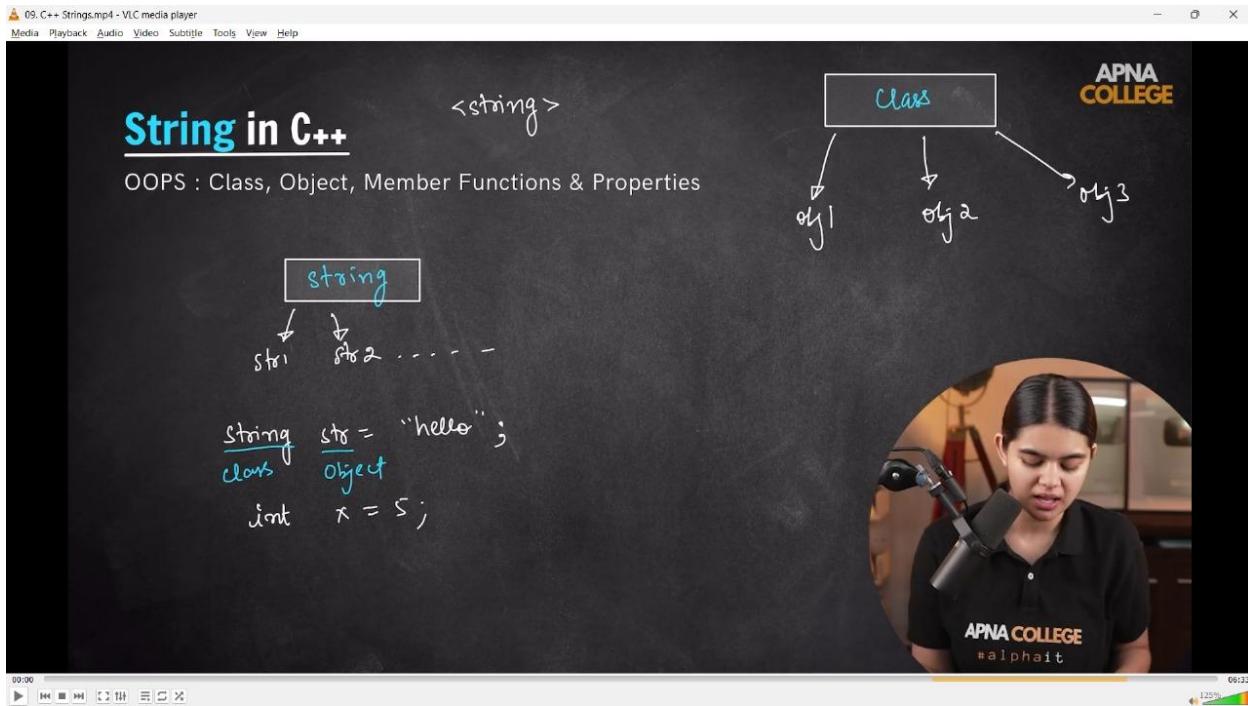
**Class**

obj1      obj2      obj3

APNA COLLEGE

APNA COLLEGE #alphait

04:30 06:18 125%



## 6.1) String Member Functions-



11. String Member Functions.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## String in C++

### Member Functions

- str.length()
- str.at( idx )
- str.substr( startIdx, size )
- str.find( word )

substring  
continuous

subarray  
continuous

"HelloWorld"

ello → str.substr( 1, 5 )



02:43 07:03 125%

APNA COLLEGE  
#alphabit

11. String Member Functions.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## String in C++

### Member Functions

- str.length()
- str.at( idx )
- str.substr( startIdx, size )
- str.find( word )

"I love C++ & Java"

str.find("C++") idx at double

str.find("Python") -1



04:40 07:03 125%

APNA COLLEGE  
#alphabit

```
// int main()
//{
//    string str = "Microsoft Hyderabad";
```

```

// cout << str.length() << endl; // 19

// cout << str[3] << endl; // r

// cout << str.at(3) << endl; // r

// cout << str.substr(5, 17) << endl; // soft Hyderabad

// string str2;

// cout << "what's in your mind baby - " << endl;

// getline(cin, str2); // Microsoft's new office in Pune

// cout << str2.find("Pune") << endl; // 26

// cout << str2.find("GIFT CITY") << endl; // 18446744083781055516154 - which is a Garbage
value and eq. to -1. Find fun returns a unsigned value which is this.

// // want to see this as -1

// int idx = str2.find("GIFT CITY");

// cout << idx << endl; // -1

// // Similarity for more than one occurrence. Just start indexing before that occurrence to
count that

// string str3;

// cout << "what's in your mind baby - " << endl;

// getline(cin, str3);

// cout << str3.find("office") << endl; // 26

// cout << str3.find("office", 30) << endl; // 53

// cout << str3.find("office", 54) << endl; // 96

// // String Operations -

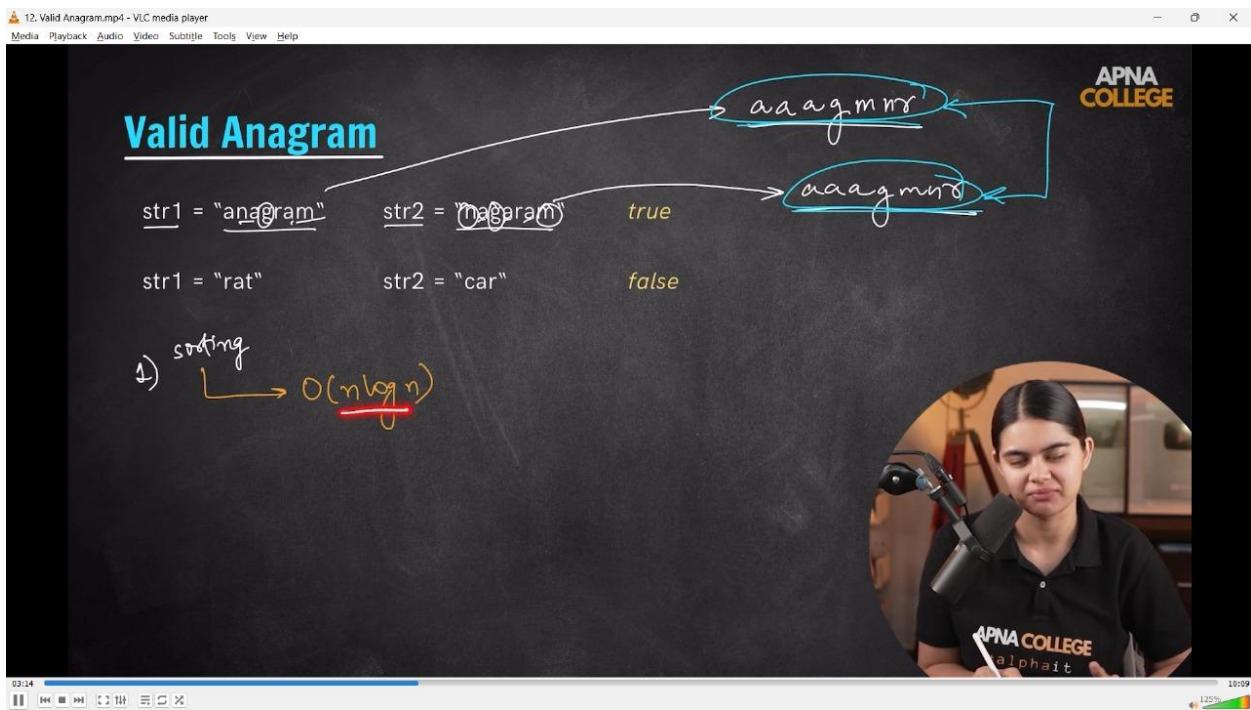
```

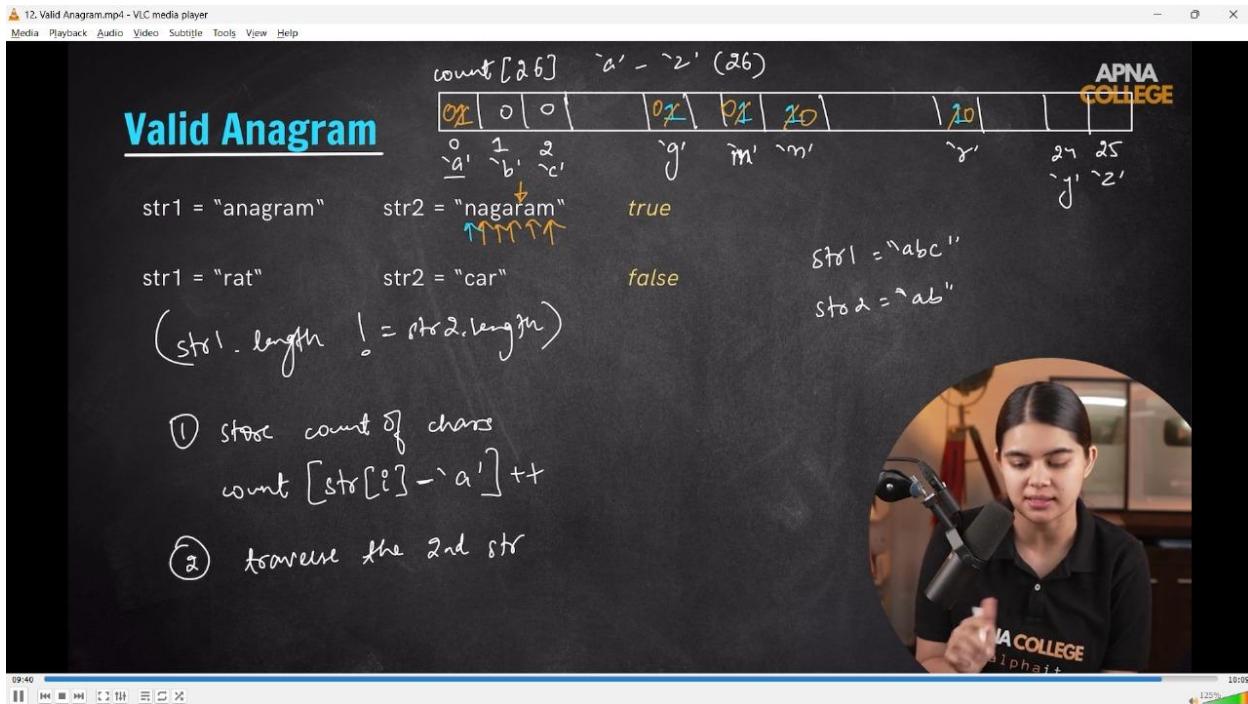
```

// string t1 = "cat";
// string t2 = "dog";

// string t1 = "cat";
// string t2 = "dog";
// cout << (t1 != t2) << endl;//1
// cout << (t1 > t2) << endl;//0
// cout << (t1 < t2) << endl;//1
// }
// 
```

## 7) Check the two strings are Anagram or not –





```
// bool isAnagram(string str1, string str2)

// {
//     if (str1.size() != str2.size())
//     {
//         cout << "not valid ANagrams" << endl;
//         return false;
//     }
// }

// int count[26] = {0}; // setting all character's count 0 initially
// for (int i = 0; i < str1.length(); i++)
// {
//     count[str1[i] - 'a']++;
// }
// for (int i = 0; i < str2.length(); i++)
// {
```

```
//     if (count[str2[i] - 'a'] == 0)
//
//     {
//         cout << "It's not a valid ANagam" << endl;
//         return false;
//     }
//
//     count[str2[i] - 'a']--;
//
// }
//
// cout << "Valid ANagrams" << endl;
//
// return true;
// }

// int main()
//{
//     string str1, str2;
//
//     cout << "value of string 1 - " << endl;
//     getline(cin, str1);
//
//
//     cout << "Similarly, value of string 2 is - " << endl;
//     getline(cin, str2);
//
//
//     isAnagram(str1, str2);
//
//     /*
//     value of string 1 -
//     anagram
//     Similarly, value of string 2 is -
//     nagaram
```

```
// Valid ANagrams

// value of string 1 -
// abcdef
// Similarly, value of string 2 is -
// dabfec
// Valid ANagrams

// value of string 1 -
// shruti
// Similarly, value of string 2 is -
// rutish
// Valid ANagrams

// value of string 1 -
// madhuri
// Similarly, value of string 2 is -
// ridhum
// not valid ANagrams

// T.C - O(n+m) or any of them which dominates as per the length of string 1 or 2
// */
// _____
```

## [6] - STL - Vector

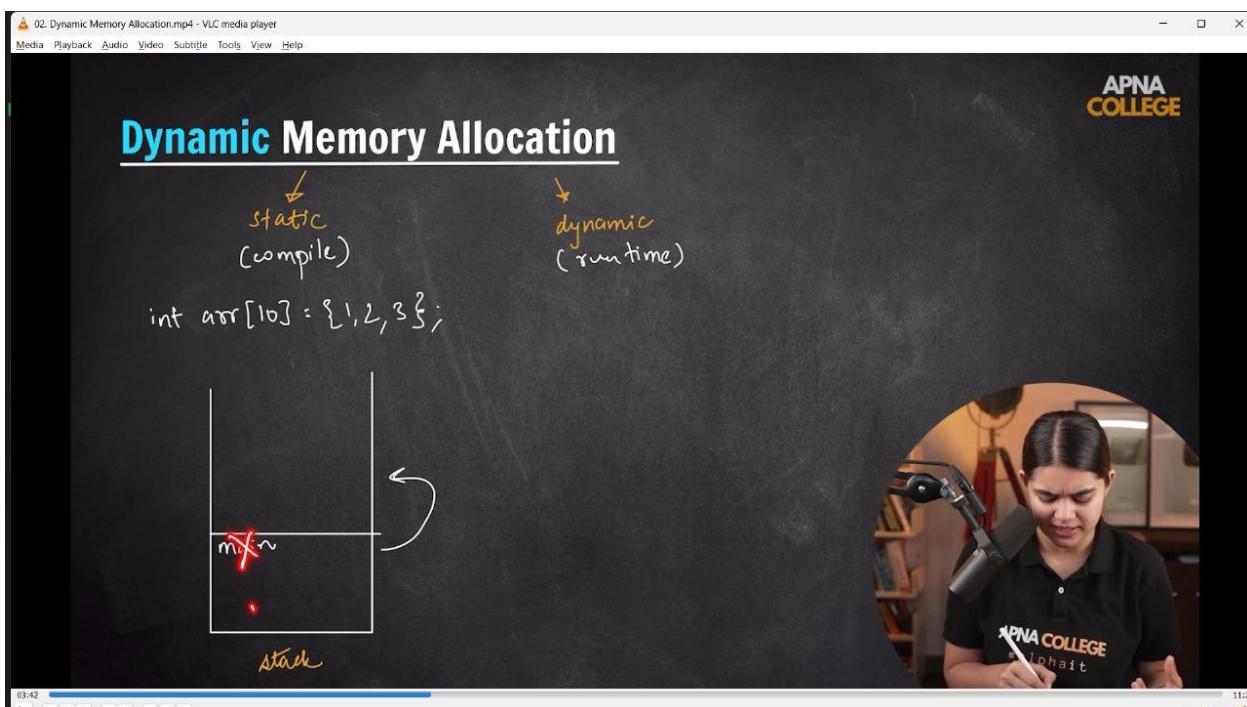
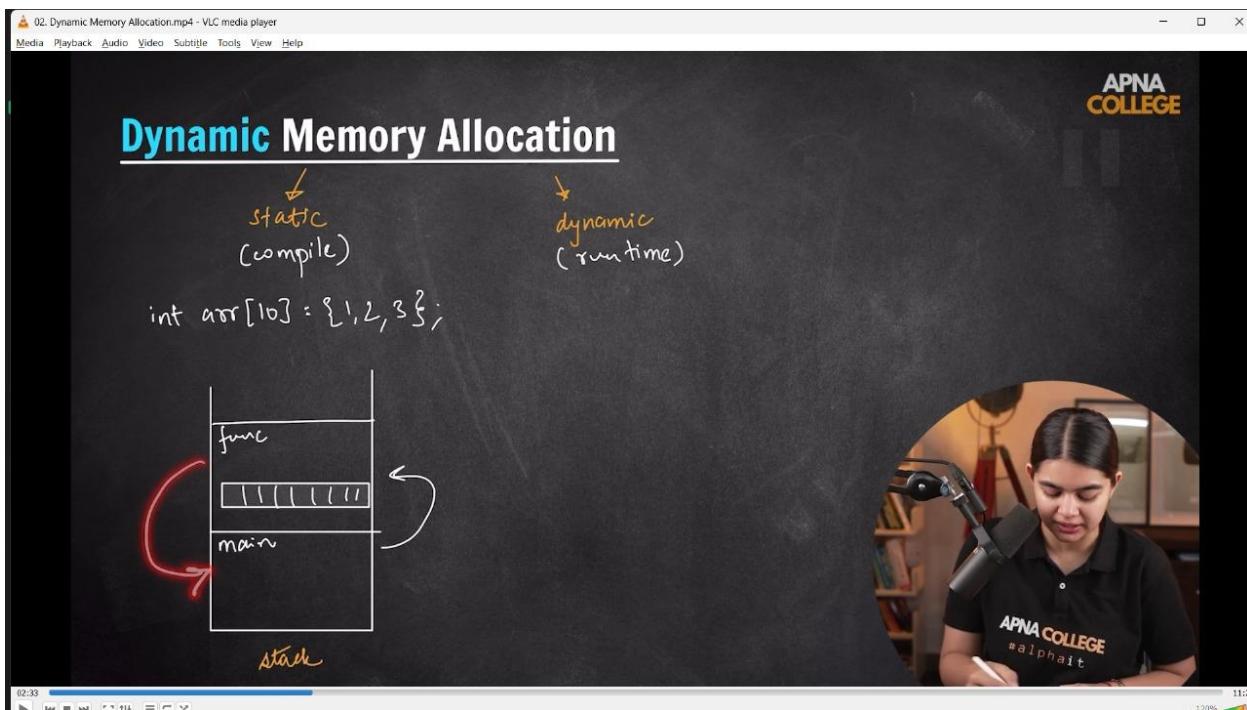
1) Pair\_Sum - Find if any pair in Sorted Array has target sum

1.1) Brute Force Approach - using Nested Loops. TC -  $O(n^2)$

1.3) Using Linear Approach - 2 Pointer Approach. TC -  $O(n)$

## 6) DMA & STL in CPP –

### //1) Dynamic Memory Allocation -



02. Dynamic Memory Allocation.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Dynamic Memory Allocation

static (compile)

dynamic (runtime)

```
int arr[10] = {1, 2, 3};
```

heap

main

stack

int \*ptr = new int[10];

APNA COLLEGE

11:24

02. Dynamic Memory Allocation.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Dynamic Memory Allocation

static (compile)

dynamic (runtime)

```
int arr[10] = {1, 2, 3};
```

heap

memory leak

ptr (ptr+2)

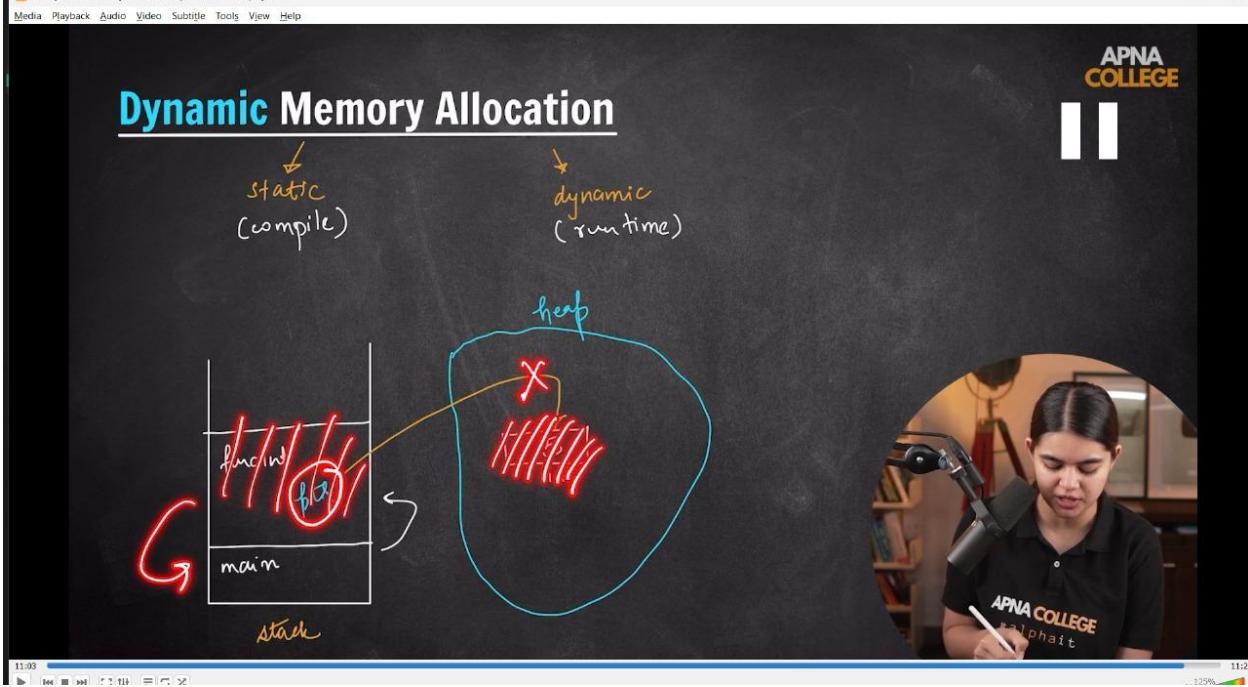
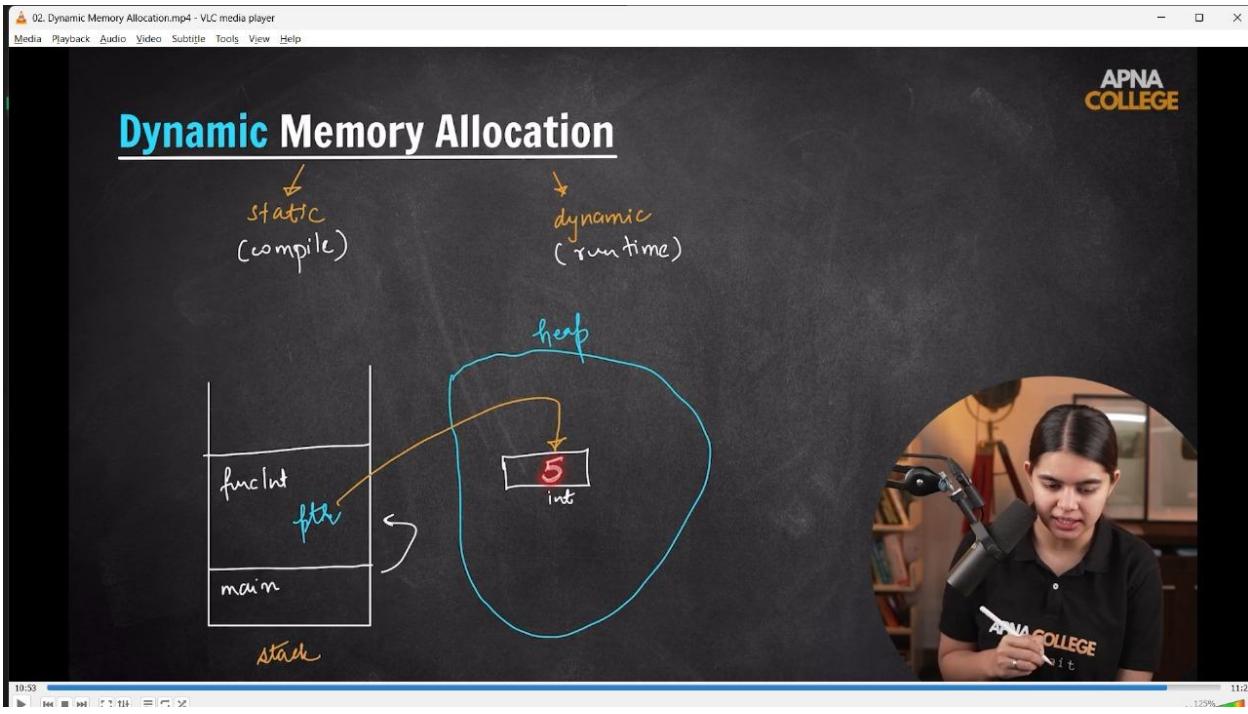
main

int \*ptr = new int[10];

ptr[2] → \*(ptr+2)

APNA COLLEGE

11:24



## Difference B/w DMA & SMA -

The slide has a dark background with the title 'Dynamic Memory Allocation' in large blue font. Below it, two sections are shown: 'Static Memory Allocation' and 'Dynamic Memory Allocation'. The 'Static Memory Allocation' section lists:

- allocation at compile time
- uses stack memory
- gets freed automatically

The 'Dynamic Memory Allocation' section lists:

- allocation at run time (new operator)
- uses heap memory
- needs to be freed explicitly (delete operator)

A circular video frame in the bottom right corner shows a person speaking into a microphone.

## Memory Leak –

The slide has a dark background with the title 'Memory Leak' in large blue font. Below it, the text states: 'A memory leak occurs when programmers create a memory in a heap and forget to delete it.' and 'Leads to reduced performance due to depletion of available memory.' To the right, handwritten notes show a pointer variable  $ptr = 100$  pointing to the start of a heap block. The heap block contains several memory cells, some of which are marked with question marks, indicating they are no longer used but still allocated.

A circular video frame in the bottom right corner shows a person holding a pen and speaking into a microphone.

```
// int main()
```

```
// {
```

```
// /*  
// // int arr[100] = {1,2,3,4,5};  
// int size;  
// cin >> size;//10  
  
// int *arr = new int[size]; // so by using new keyword we can create an arrayu dynamically in  
memory  
// arr[0] = *(arr + 0);  
// arr[1] = *(arr + 1);  
// arr[2] = *(arr + 2);  
  
// int x = 1;  
// for (int i = 0; i < size; i++)  
// {  
//     arr[i] = x;  
//     cout << arr[i] << " ";  
//     x++;  
// }  
// cout << endl;  
// /*  
// 10  
// 1 2 3 4 5 6 7 8 9 10  
// */  
// _____
```

/\*  In SMA - var,aray memo, fun me create hoti he staack me run hoti he within the fun end ho jaati he, no access in main fun or anywhere.

but in DMA - Memory HEap me create hoti he and throughout the code can be accessible from anywhere

2) SO, here using new keyword. it's heap memory, but it'll also appear every time still when no one uses it. so as a

program mandatory to delete the memory too.

3) Hence new keyword uses kr ke pointer array or var bna liya lekin after using it in the main fun it must be deleted again in the respective fun as the memory created in heap and will be existing throughout the code until deleted - then bhai hi use na ho rhi ho

\*/

```
// void funcInt()  
// {  
//     int *ptr = new int;  
//     *ptr = 5;  
//     cout<<*ptr;  
  
//     delete ptr;  
// }  
  
// void funcArray()  
// {  
//     // int arr[100] = {1,2,3,4};  
//     int size;  
//     cin>>size;
```

```
// int *arr = new int[size];// Created Dynamically Allocated Memory  
// int x = 1;  
// for (int i = 0; i < size; i++)  
// {  
//     arr[i] = x;  
//     cout << arr[i] << " ";  
//     x++;  
// }  
// cout<<endl;  
// delete[] arr;// Freeing up the sapce  
// }  
  
// int main()  
// {  
//     funcArray();  
//     /*  
//      10  
//      1 2 3 4 5 6 7 8 9 10  
//      */  
//     funcInt();  
//     // 5  
// }  
// _____  
//2) DMA in 2D Dynamic Arrays -
```

05\_2D Dynamic Arrays.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## 2D Dynamic Arrays

```
ptri = new int [cols]
```

for1 →	1	2	3
ptr2 →	4	5	6
ptr3 →	7	8	9
ptr4 →	10	11	12
⋮			
n row			

array of point

```
int *arr = new int[szie]
```

```
int **matrix = new int *[size]
```

04:01 14:05 125%

05\_2D Dynamic Arrays.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## 2D Dynamic Arrays

```
ptri = new int [cols]
```

$4 \times 3$

for1 →	1	2	3
ptr2 →	4	5	6
ptr3 →	7	8	9
ptr4 →	10	11	12
⋮			
n row			

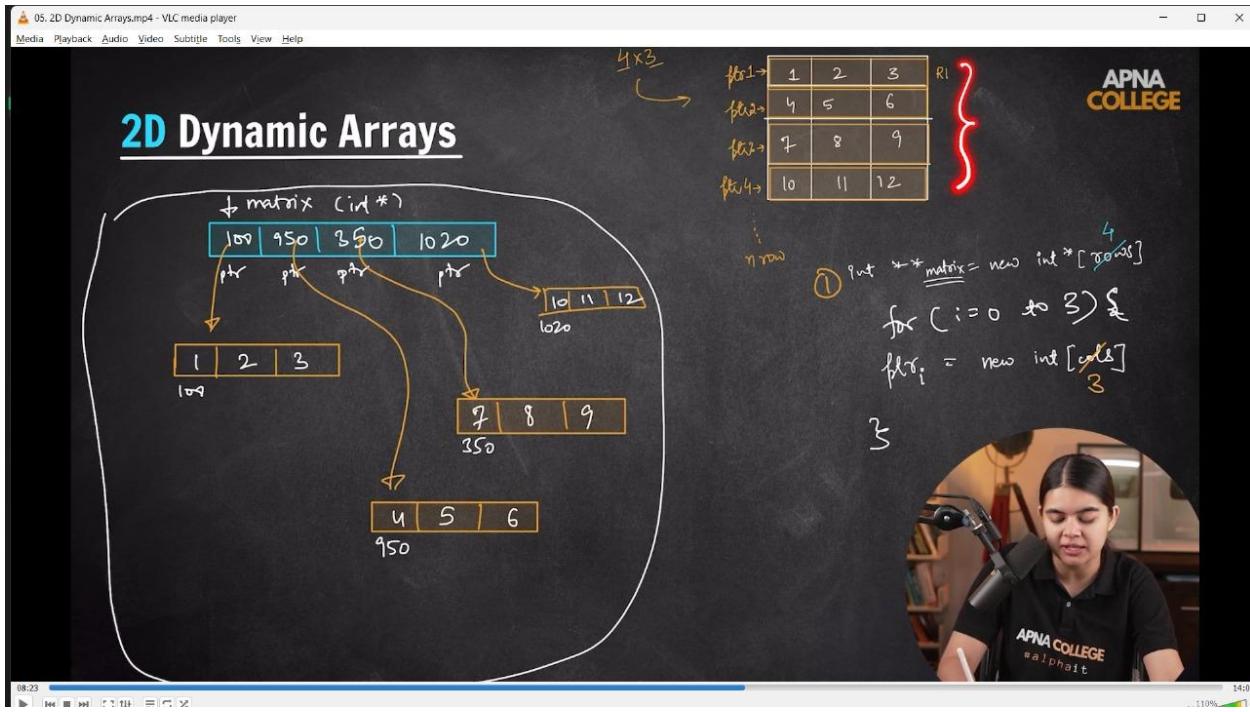
array of point

①  $\text{int } **\underline{\text{matrix}} = \text{new int } *[\underline{\text{rows}}]$

$\text{int } * \text{ptr}$

$\text{ptr} \quad \text{ptr} \quad \text{ptr} \quad \text{ptr}$

05:28 14:05 125%



```
int main()
{
    int n, m;
    cout << "No. of rows & columns are - " << endl;
    cin >> n >> m;
```

```
int **matrix = new int *[n]; // Correct syntax for 2D dynamic array
```

```
for (int i = 0; i < n; i++)
{
    matrix[i] = new int[m];
```

```
}
```

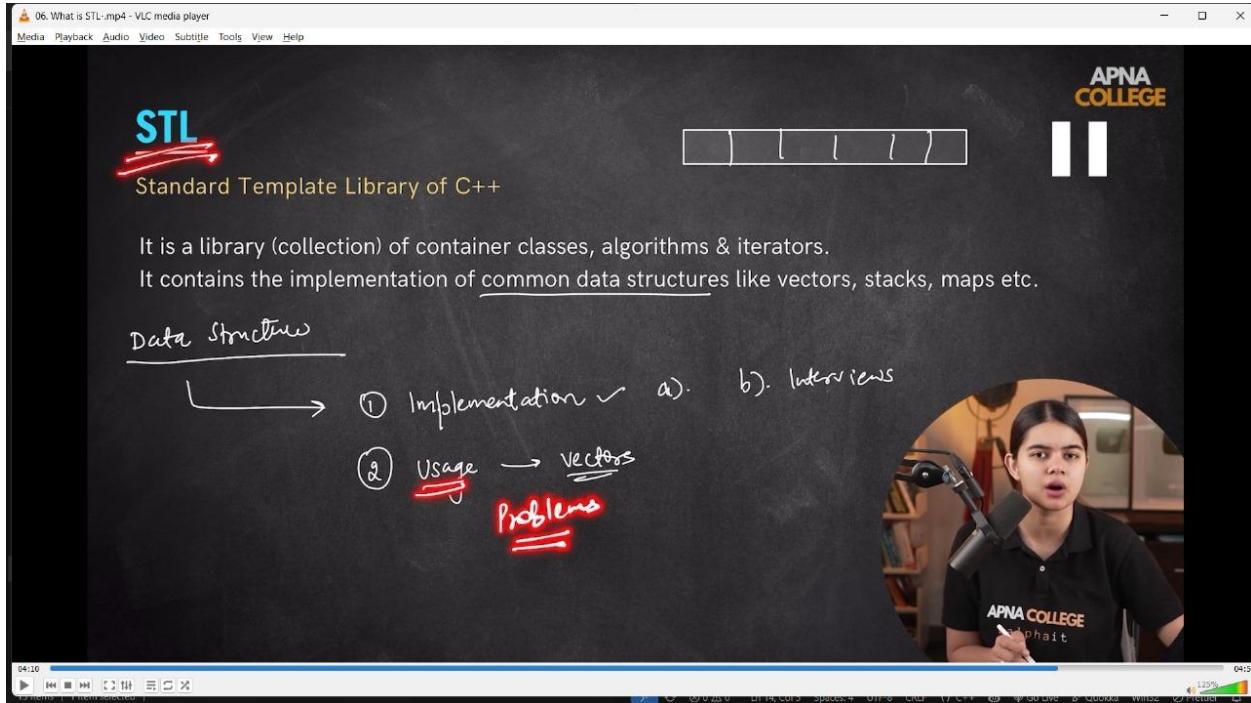
```
// Data store
```

```
int x = 1;
for (int i = 0; i < n; i++)
```

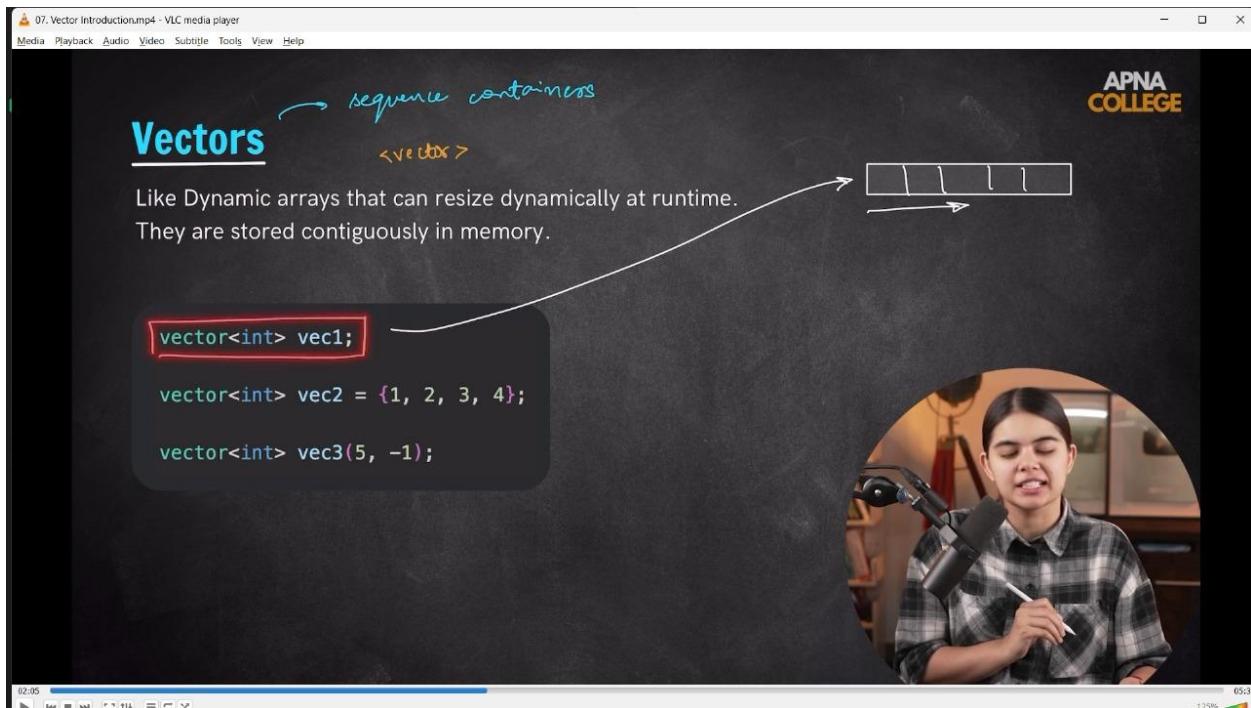
```
{  
    for (int j = 0; j < m; j++)  
    {  
        matrix[i][j] = x++;  
        cout << matrix[i][j] << " ";  
    }  
    cout << endl;  
}  
  
// Free the dynamically allocated memory  
for (int i = 0; i < n; i++)  
{  
    delete[] matrix[i];  
}  
delete[] matrix;  
  
return 0;  
/*  
No. of rows & columns are -  
3  
3  
1 2 3  
4 5 6  
7 8 9  
*/  
}
```

//

## //3)STL -



### //3.1) vectors -



07. Vector Introduction.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Vectors

→ sequence containers  
vector>

Like Dynamic arrays that can resize dynamically at runtime.  
They are stored contiguously in memory.

```

vector<int> vec1;
vector<int> vec2 = {1, 2, 3, 4};
vector<int> vec3(5, -1);
    
```

fill constructor

03:41 05:38 125%

08. Vector Implementation in Memory.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Vector Implementation in Memory

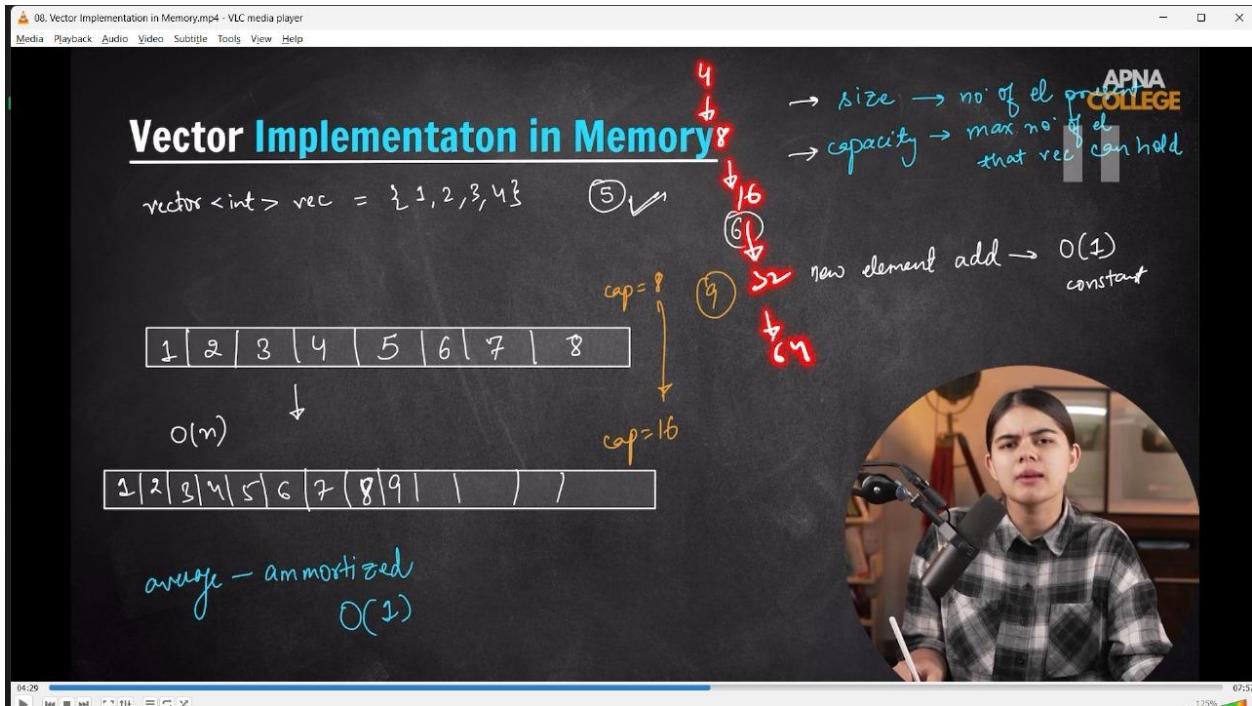
vector<int> vec = {1, 2, 3, 4} ⑤ ✓

→ size → no. of el present  
→ capacity → max no. of el that vec can hold

new element add → O(1) constant

average - amortized O(1)

04:45 07:57 125%



```
// int main()
{
    // Vector Initialization methods -
    // vector<int> vec1;
    // cout << vec1.size() << endl; // 0
    // vector<int> vec2 = {2, 4, 6, 8};
    // cout << vec2.size() << endl; // 4
    // vector<int> vec3(10, -1);
    // for (int i = 0; i < vec3.size(); i++)
    // {
    //     cout << vec3[i] << " ";
    // }
    // cout << endl;

    // cout << vec3.size() << endl; // -1 -1 -1 -1 -1 -1 -1 -1 -1
}
```

```
// cout << endl;

// /*
//   Vector Implementation in Memory - In vector it also deals with derference [ointer
// concepts woithin the memory

// 2 - Vector capacity becomes double of itslef on adding new element

// */

// vector<int> vec = {1, 2, 3, 4};

// cout << "Size is - " << vec.size() << endl;      // Size is - 4

// cout << "Capacity is - " << vec.capacity() << endl; // Capacity is - 4

// cout << endl;

// vec.push_back(5);

// cout << "Size is - " << vec.size() << endl;      // Size is - 5

// cout << "Capacity is - " << vec.capacity() << endl; // Capacity is - 8

// vec.pop_back();

// for (auto i: vec)

// {

//     cout<<vec[i]<<" ";

// }

// cout<<endl;

// }

// _____
```

```
// 3.2) Pair_Sum - Find if any pair in Sorted Array has target sum

// 3.2.1) Brute Force Approach - using Nested Loops. TC - O(n^2)

// int main()

// {

//     int n, target;

//     cout << "vector size - " << endl;

//     cin >> n;

//     vector<int> arr(n);

//     cout << "Cvecrto Eleemnts are - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << "So, inserted elements of vector are - " << endl;

//     for (auto i : arr)

//     {

//         cout << i << " ";

//     }

//     cout << endl;

//     cout << "target element is - " << endl;

//     cin >> target;

//     for (int i = 0; i < n - 1; i++)

//     {
```

```

//     for (int j = i + 1; j < n; j++)
//     {
//         if (arr[i] + arr[j] == target)
//         {
//             cout << arr[i] << " " << arr[j] << endl;
//             cout << "Indices: " << i << " " << j << endl;
//         }
//     }

// }

// return 0;

// /*
// PS E:\51 LPA_Shagun\_Level-Up-Data-Structures> cd "e:\51 LPA_Shagun\_Level-Up-Data-Structures\6_Vectors\" ; if ($?) { g++ 2_STL.cpp -o 2_STL } ; if ($?) { .\2_STL }

// vector size -
// 5

// Cvecrtor Eleemnts are -
// 10 12 8 2 5

// So, inserted elements of vector are -
// 10 12 8 2 5

// target element is -
// 18

// 10 8

// Indices: 0 2

// TC- O(N^2)

```

```
// */  
// }  
// _____
```

### //3.2.2) Using Linear Approach - 2 Pointer Approach. TC - O(n)

09. Pair Sum.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Pair Sum**

Find if any pair in sorted vector has target sum.

input : arr = [ 2, 7, 11, 15 ], target = 9

output : [0, 1] // vector of indices  
 ↑ |

APNA COLLEGE

01:00 11:30 125%

09. Pair Sum.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Pair Sum**

2 Pointer Approach

target = 9

arr = [ 2, 7, 11, 15 ], target = 9

currSum = 17

st      end

① currSum = target  $\Rightarrow$  ans

② currSum > target  
 end --

③ currSum < target  
 st ++

APNA COLLEGE

05:57 11:30 125%

09. Pair Sum.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Pair Sum

2 Pointer Approach

arr = [ 2, 7, 11, 15 ], target = 9

$\uparrow \quad \uparrow$   
st end

Diagram showing an array [ 2 | 7 | 11 | 15 ] with indices st=0 and end=3. A red circle highlights index 11. Red arrows point from 'currSum' to 'target' and from 'end' to 'st'. A curved arrow points from 'currSum' to 'ans'. The text 'ans = 9' is written above the array.

APNA COLLEGE

00:49 11:30 125%

09. Pair Sum.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Pair Sum $\underline{O(n)}$

2 Pointer Approach

arr = [ 2, 7, 11, 15 ], target = 9

$\uparrow \quad \uparrow$   
st end

Diagram showing an array [ 2 | 7 | 11 | 15 ] with indices st=0 and end=3. A red circle highlights index 11. Red arrows point from 'currSum' to 'target' and from 'end' to 'st'. A curved arrow points from 'currSum' to 'ans'. The text 'ans = 9' is written above the array.

APNA COLLEGE

00:17 11:30 125%

```
// vector<int> pairSum(vector<int> vec, int target)
//{
//    int st = 0, end = vec.size() - 1;
//    int currSum = 0;
```

```
// vector<int> Ans;

// while (st < end)
// {
//     currSum = vec[st] + vec[end];
//     if (currSum == target)
//     {
//         Ans.push_back(st);
//         Ans.push_back(end);
//         return Ans;
//     }
//     else if (currSum > target)
//     {
//         end--;
//     }
//     else
//     {
//         st++;
//     }
// }
// return Ans;
// }
```

```
// int main()
//{
//    int n;
```

```
// cout << "What is the vector Size here -" << endl;
// cin >> n;

// vector<int> vec(n);

// cout << "What are the Vector Elements - " << endl;
// for (int i = 0; i < vec.size(); i++)
// {
//     cin >> vec[i];
// }

// cout << "Hence, the inserted vector is - " << endl;
// for (auto i : vec)
// {
//     cout << i << " ";
// }
// cout << endl;

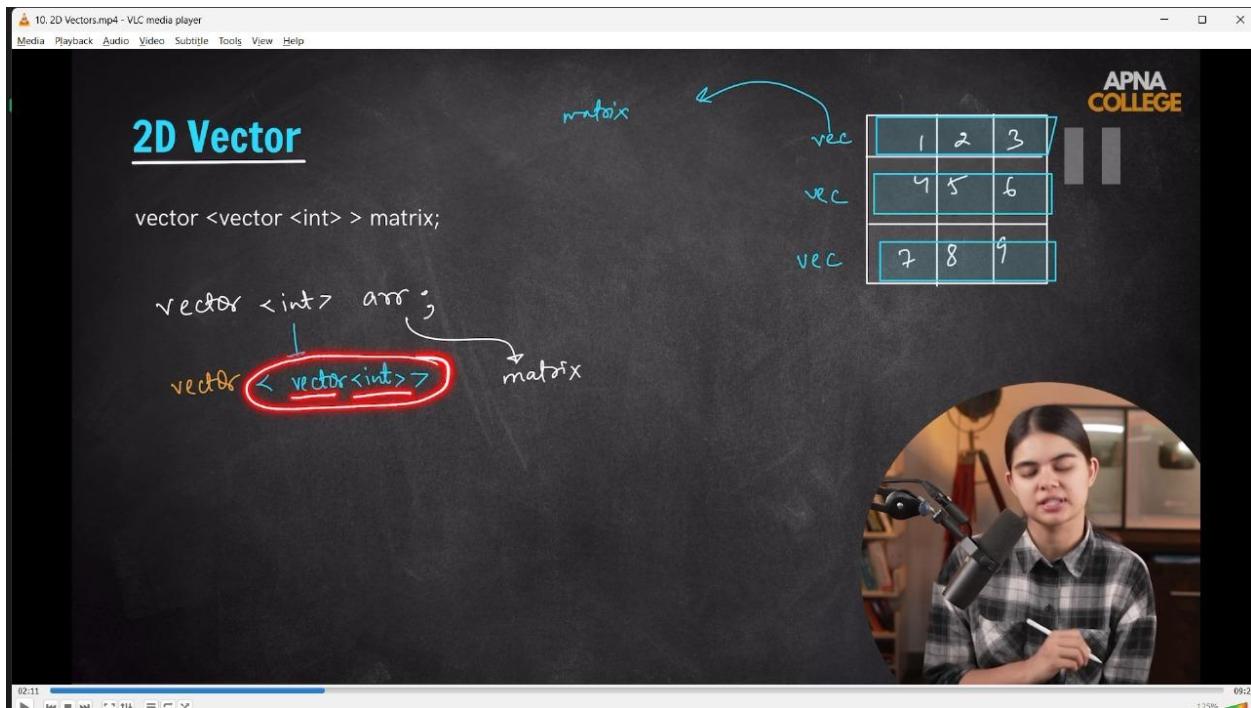
// int target;
// cout << "Mention the value you want to target - " << endl;
// cin >> target;

// vector<int> Ans = pairSum(vec, target);

// cout << "The indices of the elements that sum to target are: ";
// for (auto idx : Ans)
// {
```

```
//      cout << idx << " ";
//  }
//  cout << endl;
// /*
// What is the vector Size here -
// 4
// What are the Vector Elements -
// 2 7 11 15
// Hence, the inserted vector is -
// 2 7 11 15
// Mention the value you want to target -
// 9
// The indices of the elements that sum to target are: 0 1
// -----
// What is the vector Size here -
// 4
// What are the Vector Elements -
// 2 7 11 15
// Hence, the inserted vector is -
// 2 7 11 15
// Mention the value you want to target -
// 17
// The indices of the elements that sum to target are: 0 3
// */
// }
```

## // 4) 2-D Vector -



```
// int main()
// {
//     vector<vector<int>> Matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
//
//     for (int i = 0; i < Matrix.size(); i++)
//     {
//         for (int j = 0; j < Matrix[i].size(); j++)
//         {
//             cout << Matrix[i][j] << " ";
//         }
//         cout << endl;
//     }
// /*
```

```
// 1 2 3
// 4 5 6
// 7 8 9
// */
/// -----  

// // In 2D Vector no issue of assigning row and colm. It can be done in any way easily
// vector<vector<int>> Matrix1 = {{1, 2, 3}, {4, 5}, {6}};  

// for (int i = 0; i < Matrix1.size(); i++)
// {
//     for (int j = 0; j < Matrix1[i].size(); j++)
//     {
//         cout << Matrix1[i][j] << " ";
//     }
//     cout << endl;
// }
// /*
// 1 2 3
// 4 5
// 6
// */
/// -----  

// // Similarly by trying from taking the input from the user -
// int n, m;
// cout << "Enter the number of rows & columns respectively: ";
```

```
// cin >> n >> m;

// vector<vector<int>> M2(n, vector<int>(m)); // Proper 2D vector initialization

// cout << "Mention the vector elements:" << endl;
// for (int i = 0; i < n; i++)
// {
//     for (int j = 0; j < m; j++)
//     {
//         cin >> M2[i][j];
//     }
// }

// cout << "The inserted elements of the vector are:" << endl;
// for (int i = 0; i < n; i++)
// {
//     for (int j = 0; j < m; j++)
//     {
//         cout << M2[i][j] << " ";
//     }
//     cout << endl;
// }

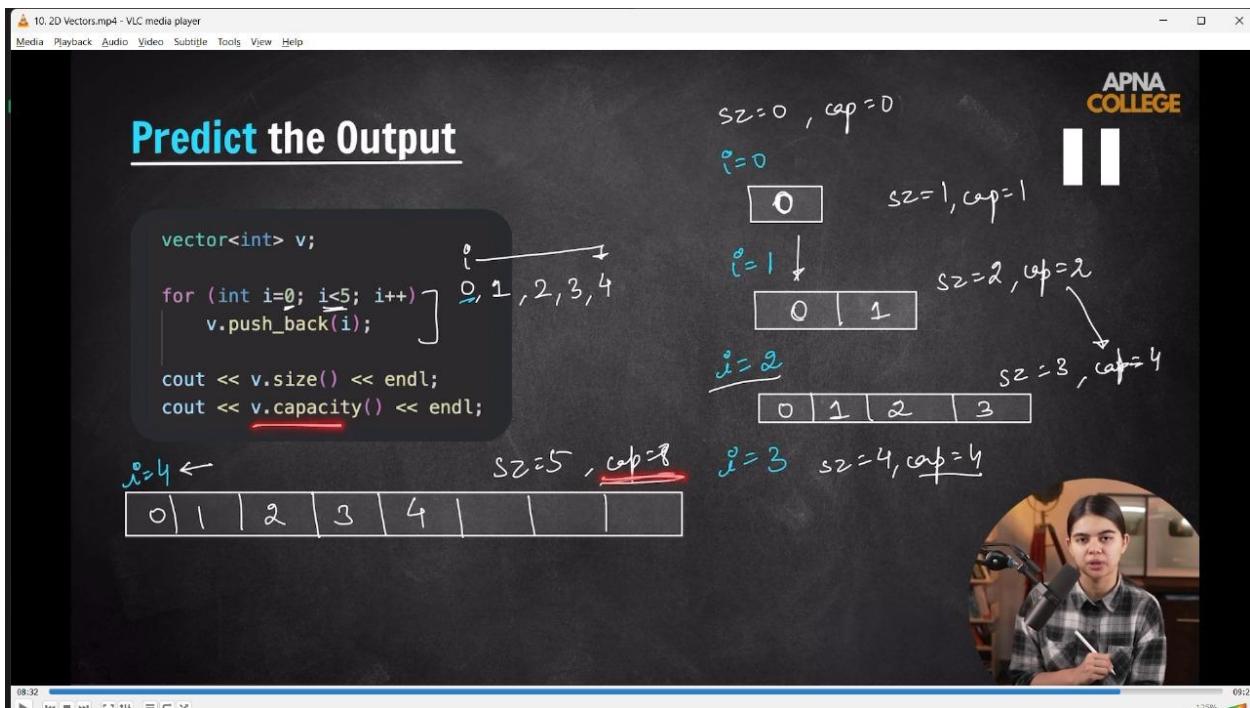
// /*
// Enter the number of rows & columns respectively: 4 4
// Mention the vector elements:
// 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

```
// The inserted elements of the vector are:
```

```
// 1 2 3 4  
// 5 6 7 8  
// 9 10 11 12  
// 13 14 15 16  
// */  
// }
```

---

```
// Qn) What will be the o/p -
```



```
// int main()  
// {  
//     vector<int> v;  
//     for (int i = 0; i < 5; i++)  
//     {  
//         v.push_back(i);
```

```
// }  
// cout<<v.size()<<endl;// 5  
// cout<<v.capacity()<<endl;// 8  
// }
```

---

---

## **[7] Bit Manipulations –**

- 1)Bitwise Operators -
- 2) Check if even or odd -
- 3) Get ith Bit of a num -
- 4) Set ith Bit - (sets the value to 1) -
- 5)Clr ith Bit - (sets the value to 0)
- 6) No. is power of 2 -
- 7) Count of Set Bits -
- 8) Fast Exponentiation -

## **7) Bit Manipulations –**

## 1)Bitwise Operators -

01. Bitwise Operators.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

### Bitwise Operators

- Bitwise AND &
- Bitwise OR |
- Bitwise XOR ^

Rule

$$\begin{array}{l} 0 \& 1 \rightarrow 0 \\ 0 \& 0 \rightarrow 0 \\ 1 \& 0 \rightarrow 0 \\ 1 \& 1 \rightarrow 1 \end{array}$$

$$\begin{array}{r} 3 \& 5 \\ \hline 00011 \\ \& 00101 \\ \hline 00001 \end{array} = (1)_{10}$$

APNA COLLEGE

04:10 67:35 125%

01. Bitwise Operators.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

### Bitwise Operators

- Bitwise AND &
- Bitwise OR |
- Bitwise XOR ^

Rule

$$\begin{array}{l} 0 | 0 \rightarrow 0 \\ 0 | 1 \rightarrow 1 \\ 1 | 0 \rightarrow 1 \\ 1 | 1 \rightarrow 1 \end{array}$$

$$\begin{array}{r} 3 | 5 \\ \hline 00111 \\ \& 01011 \\ \hline 01111 \end{array} = (7)_{10}$$

APNA COLLEGE

05:56 67:35 125%

01. Bitwise Operators.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Bitwise Operators

- Bitwise AND &
- Bitwise OR |
- Bitwise XOR ^
  - exclusive OR

Rules

same → 0  
diff → 1

$$\begin{array}{rcl} 3 \wedge 5 & & \\ 0011 & & \\ \wedge & \overline{0101} & \\ \hline 0110 & = (6)_{10} & \end{array}$$


APNA COLLEGE

07:23 07:35 125%

02. One's Complement.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## One's Complement

Binary NOT Operator

$\sim \rightarrow \text{Hilde}$

$$\begin{array}{l} \text{num=6} \quad \rightarrow \quad 110 \\ \underline{\sim 6} \quad \quad \quad \downarrow \downarrow \downarrow \\ \quad \quad \quad 001 \quad \Rightarrow 1_x \\ \quad \quad \quad (-7) \end{array}$$


APNA COLLEGE

03:18 11:11 125%

02. One's Complement.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## One's Complement

Binary NOT Operator

(8 bits)      num = 6 →       $\underline{\underline{00000110}}$       APNA  
 $\underline{\underline{11111001}}$       1's complement form

$\textcircled{1} \underline{\underline{1\ 1\ 1\ 1\ 0\ 0\ 1}}$       MSB      (-7)

① 1's complement

0 0 0 0 1 1 0.  
+ 1.  
-----  
0 0 0 0 1 1 1 = (-7)<sub>10</sub>

magnitude

02. One's Complement.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## One's Complement

Binary NOT Operator

(8 bits)      num = 6 →       $\underline{\underline{00000110}}$       APNA  
 $\underline{\underline{11111001}}$       1's complement form

$\textcircled{1} \underline{\underline{1\ 1\ 1\ 1\ 0\ 0\ 1}}$       MSB      (-7)

① 1's complement

0 0 0 0 1 1 0.  
+ 1.  
-----  
0 0 0 0 1 1 1 = (-7)<sub>10</sub>

magnitude

32 bits  
↓  
1 MSB      31 bits (mag)  
2's complement

02. One's Complement.mp4 - VLC media player

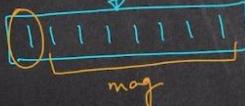
Media Playback Audio Video Subtitle Tools View Help

## One's Complement

Binary NOT Operator

$\sim 0$  Bit  $\Rightarrow 1$

$\sim 0$   $\Rightarrow 00000000$   $\downarrow$   
8 bits

$\sim 0$   $\Rightarrow$  

$\sim 0 = -1$

$11111111$   
 $00000000$   
 $+ 1$   
 $00000001 = (1)_{10}$

prog -ve num  
 $\downarrow$   
 2's complement of num

APNA COLLEGE



10:38 11:11 125%

03. Bitwise Shift Operators.mp4 - VLC media player

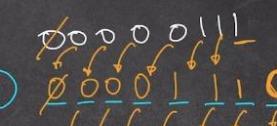
Media Playback Audio Video Subtitle Tools View Help

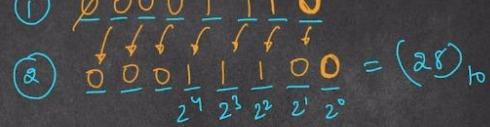
## Binary Shift Operators

- Left Shift  $<<$
- Right Shift  $>>$

$7 << 2 =$

num  $<< 2 =$  APNA COLLEGE

①   
 $00000001 \leftarrow \leftarrow$   
 $00000010 = (2^1)_{10}$

②   
 $00000001 \leftarrow \leftarrow$   
 $00000010 = (2^1)_{10}$

$16 + 8 + 4 = 28$

APNA COLLEGE



03:18 07:11 125%

03. Bitwise Shift Operators.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Binary Shift Operators

• Left Shift  $<<$

• Right Shift  $>>$

$a << b \Rightarrow a * 2^b$

$7 << 2 \Rightarrow 7 * 2^2 = 28$

APNA COLLEGE

04:42 07:11 125%

03. Bitwise Shift Operators.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Binary Shift Operators

• Left Shift  $<<$

• Right Shift  $>>$

$7 \Rightarrow 2^2$

$\boxed{0\ 0\ 0\ 0\ 0\ 0\ 1\ 1}$

$\boxed{\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{array}}$

$\boxed{\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}}$

APNA COLLEGE

05:58 07:11 125%

03. Bitwise Shift Operators.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

APNA COLLEGE

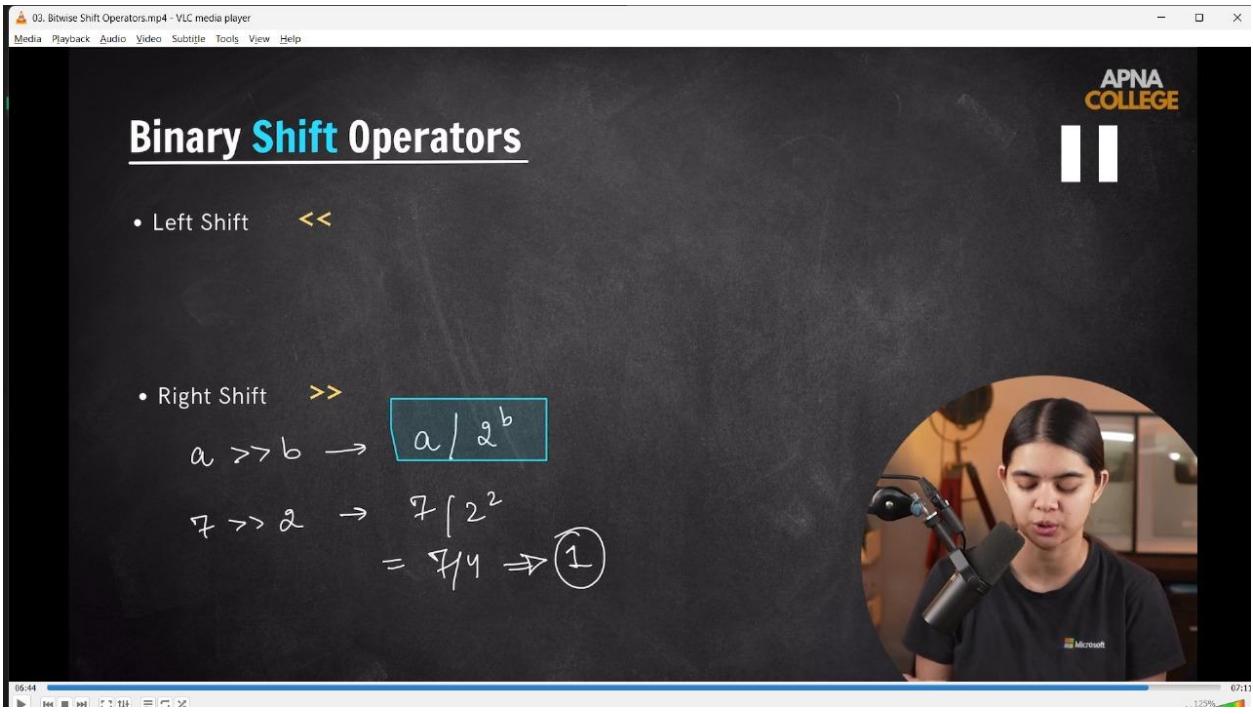
Binary Shift Operators

• Left Shift       $<<$

• Right Shift       $>>$

$a >> b \rightarrow a / 2^b$

$7 >> 2 \rightarrow 7 / 2^2$   
 $= 7 / 4 \Rightarrow 1$



04. Practice Qs.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Practice Qs

Qs. Predict the output for ~4

$\sim 4$

$\downarrow$

$00000100$

$\boxed{1111011}$

MSB

mag

(-5)

$\rightarrow$

$0000100$

$+ 1$

$0000101$

$= (5)_{10}$

APNA COLLEGE

01:04 02:12 125% Continue X

Do you want to restart the playback where left off?

A woman is speaking into a microphone.

01:04 02:12 125% Continue X

## Practice Qs

Qs. Predict the output for ~4

$\sim 4$

$\downarrow$

$00000100$

$\boxed{1111011}$

MSB

mag

(-5)

$\rightarrow$

$0000100$

$+ 1$

$0000101$

$= (5)_{10}$

APNA COLLEGE

a >> b → a / 2<sup>b</sup> ⇒ 8 / 2<sup>1</sup> ⇒ 4

8 >> 1

$000001000$

$\rightarrow (4)_{10}$

A woman is speaking into a microphone.

```
// int main()
{
    // // Bitwise AND
    cout << (3 & 5) << endl; //(0001)&(00101) = (00001) = 1
```

```
// // Bitwise OR
// cout << (3 | 5) << endl; //(0001)|(00101) = (0111) = 7

// // Bitwise XOR
// cout << (3 ^ 5) << endl; //(0001)^ (00101) = (0110) = 6 - Same bit =0, diff. bit = 1

// // Bitwise NOT(tild operator)
// cout << (~6) << endl; // -7

// cout << (~0) << endl; // -1

// // Binary Shift Operator -
// // Binary Left Shift Operator -
// cout << (7 << 2) << endl; // 28 - shortcut is simple. If a<<b then = a*2^b

// cout << (7 >> 2) << endl; // 1 - shortcut is simple. If a>>b then = a/2^b

// // Some Practice Qns -
// // 1) Output for ~4 -
// cout << (~4) << endl; // -5

// // 2) Output for - 8>>1
// cout << (8 >> 1) << endl; // 4

// }
```

// \_\_\_\_\_

## 2) Check if even or odd -

05. Check if Odd or Even.mp4 - VLC media player

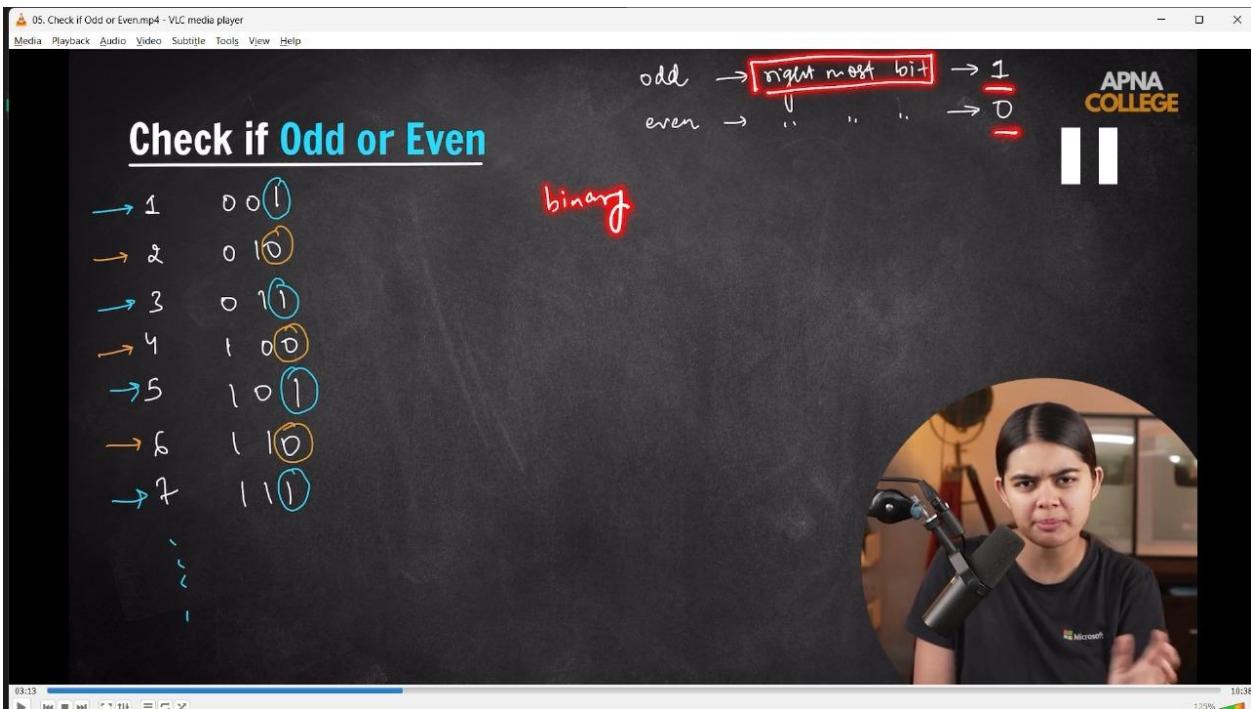
Media Playback Audio Video Subtitle Tools View Help

odd → right most bit → 1  
even → ... → 0

**Check if Odd or Even**

binary

→ 1	0 0 1
→ 2	0 1 0
→ 3	0 1 1
→ 4	1 0 0
→ 5	1 0 1
→ 6	1 1 0
→ 7	1 1 1
⋮	⋮
⋮	⋮



05. Check if Odd or Even.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

odd → right most bit → 1  
even → ... → 0

**Check if Odd or Even**

Bitmask

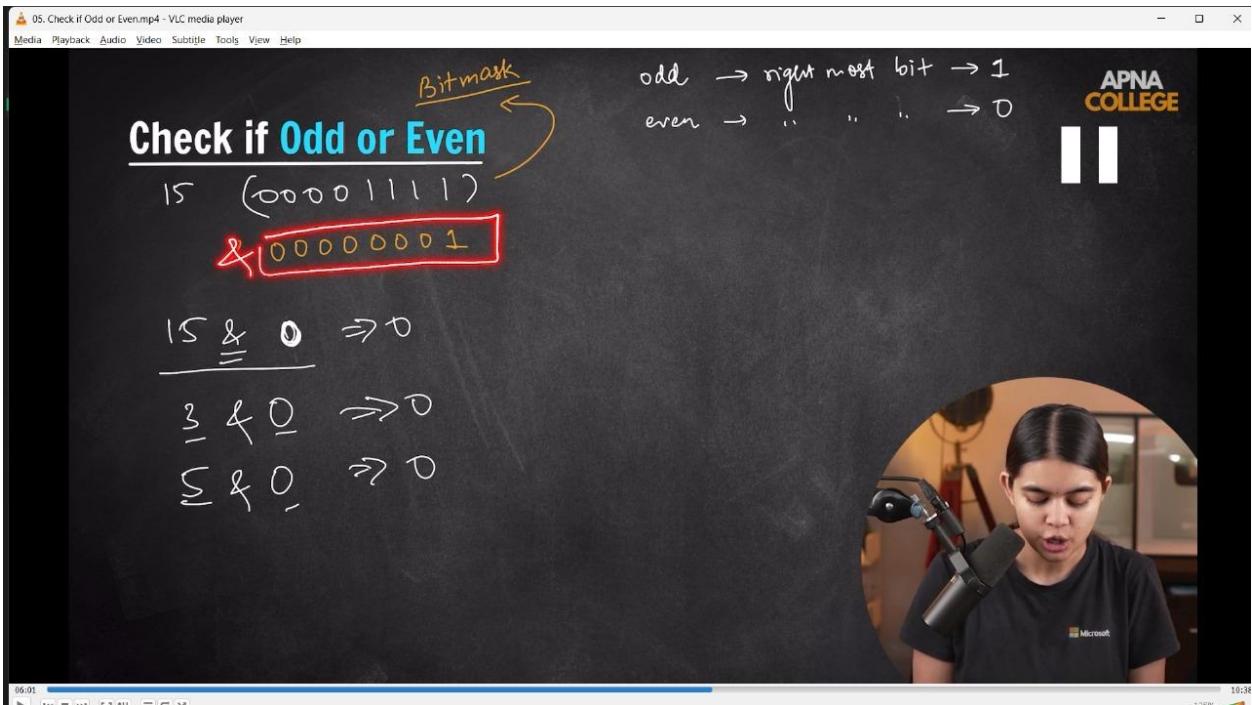
15 (00001111)

& 1000000001

$15 \& 0 \Rightarrow 0$

$3 \& 0 \Rightarrow 0$

$5 \& 0 \Rightarrow 0$



05. Check if Odd or Even.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Check if Odd or Even**

Bitmask

15  $(00001111)$

$\& [00000001] (1)_{10}$

odd  $\rightarrow$  right most bit  $\rightarrow 1$   
even  $\rightarrow \dots \dots \rightarrow 0$

$ans = 0 \Rightarrow \text{even}$   
 $\neq 0 \Rightarrow \text{odd}$

6 & 1  $\rightarrow$  110  
 $\& 001$   
 $\underline{\quad\quad\quad}$   
 $\quad\quad\quad 000$

8 & 1  $\rightarrow$  1000  
 $\& 0001$   
 $\underline{\quad\quad\quad}$   
 $\quad\quad\quad 0000$  ✓

10:25 10:38 125%

05. Check If Odd or Even.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Check if Odd or Even**

Bit masks are used to access specific bits in a byte of data.

Bitmasking

10:18 10:38 125%

/\* // Logic - In the last of every odd no in Binary(right most bit) form it consist 1.

while in the last of every even no in Binary number(right most bit) it cobsist 0.

So, for check even odd just need to chck the last bit is 0 or 1?

It could be done using (&) Bitwise AND Operator.

Any No. & 0 = will always 0.

Any No. | 0 = Will always give that no.

Any Even No. & 1 = Will alyas 0

Any Odd No. & 1 != 0

```
/*
// void oddOrEven(int num)
//{
//    if ((num & 1) == 0)// here this 1 with AND is known as BitMask - are used to access specific bits in a byte of data
//    {
//        cout << "Clearly, it's an Even Number" << endl;
//    }
//    else
//    {
//        cout << "Ohh No, it's an Odd Number dear" << endl;
//    }
//}
// int main()
//{
//    int num;
//    cout << "Enter the number you want to check for Even or Odd - " << endl;
//    cin >> num;
//    oddOrEven(num);
//}
```

```
// Enter the number you want to check for Even or Odd -
```

```
// 12
```

```
// Clearly, it's an Even Number
```

```
// Enter the number you want to check for Even or Odd -
```

```
// 15
```

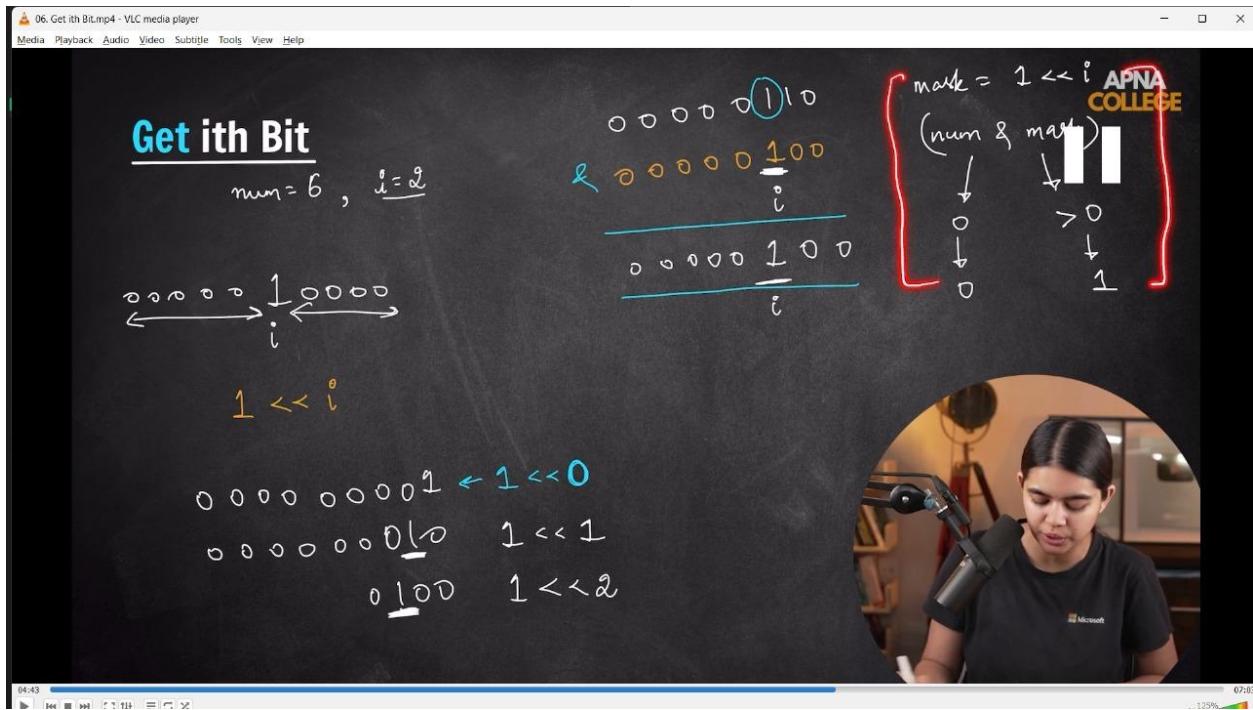
```
// Ohh No, it's an Odd Number dear
```

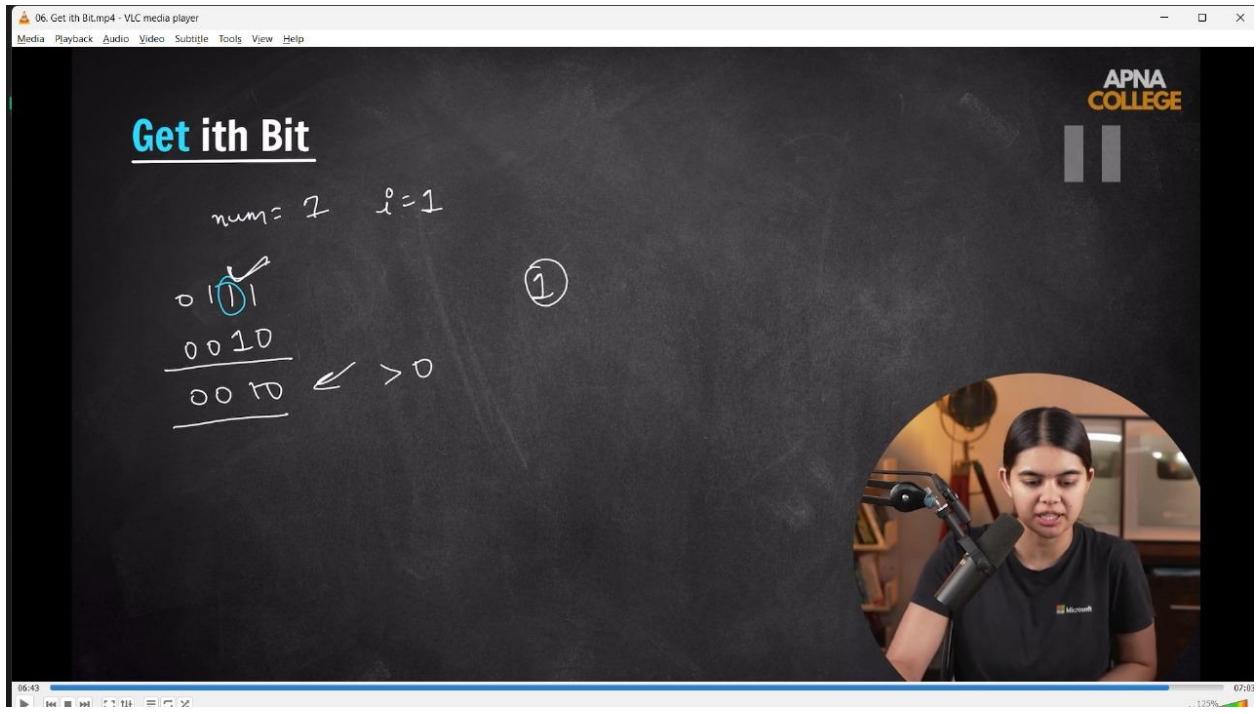
```
// */
```

```
// }
```

```
// _____
```

### 3) Get ith Bit of a num





```
// int getIthBit(int num, int i)
```

```
// {
```

```
//   int BitMask = 1 << i;
```

```
//   if (!(num & BitMask))
```

```
//   {
```

```
//     return 0;
```

```
//   }
```

```
// else
```

```
// {
```

```
//   return 1;
```

```
// }
```

```
// }
```

```
// int main()
```

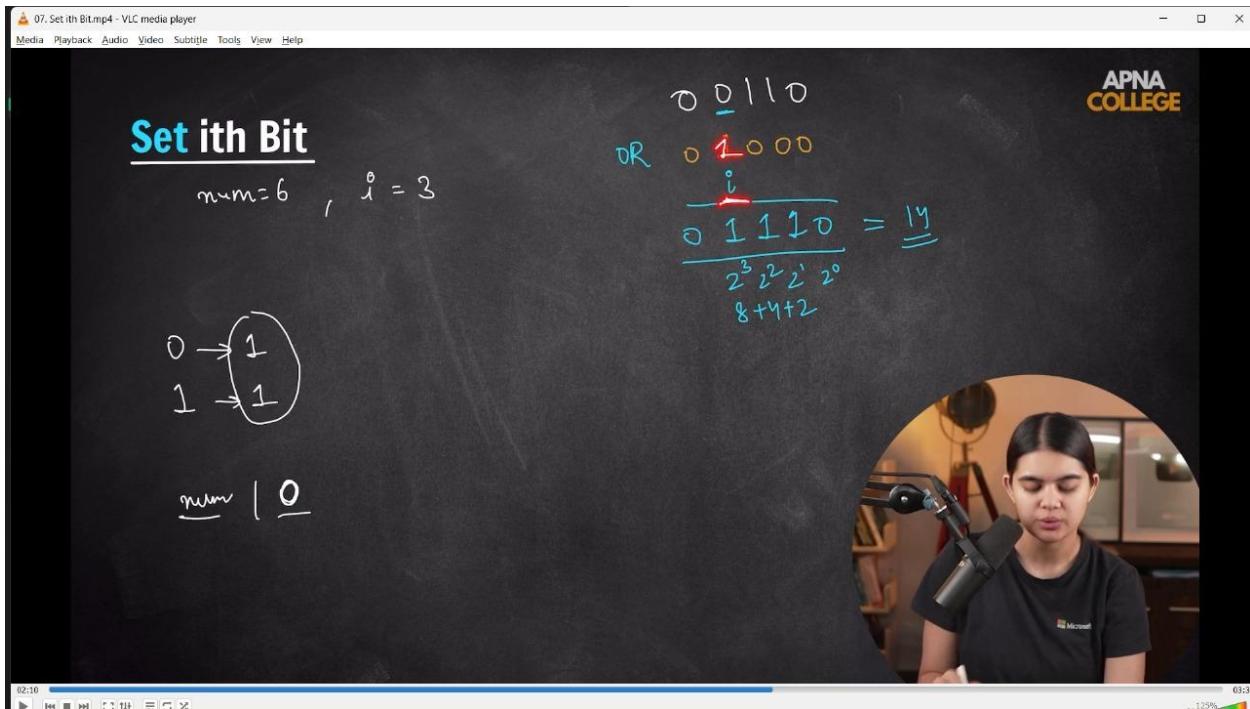
```
// {
```

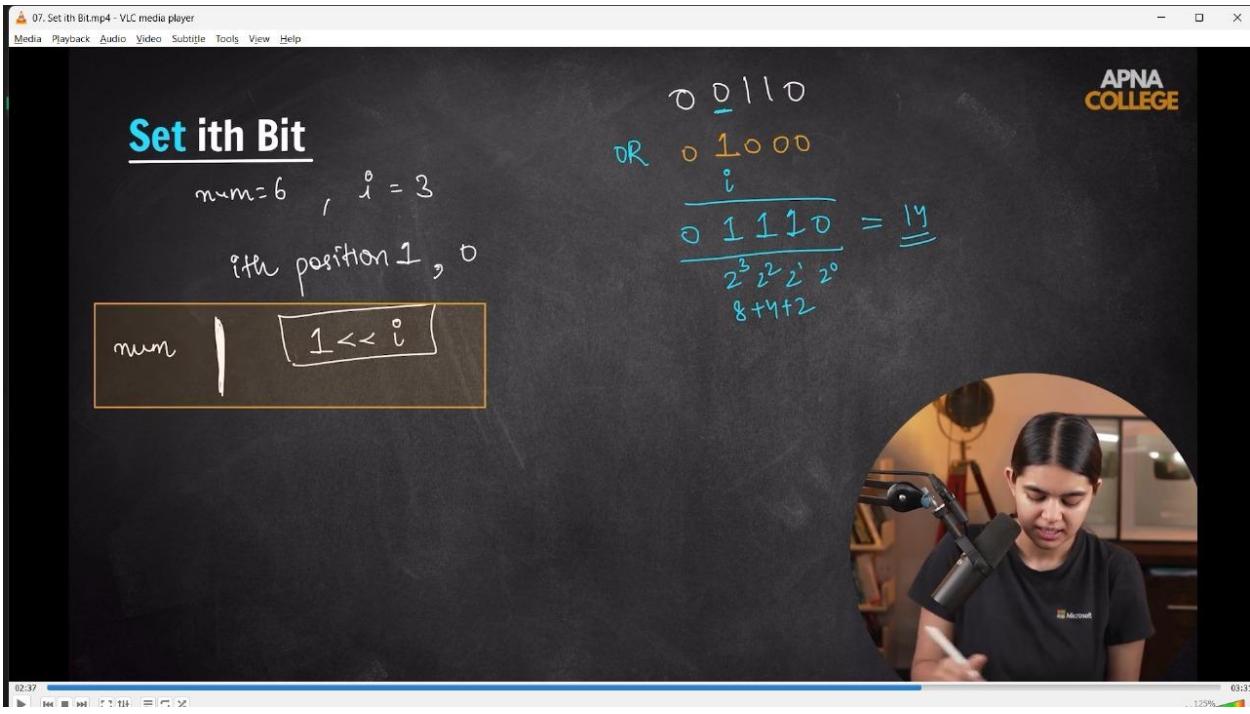
```

// cout << getIthBit(6, 2) << endl; // 1 - in BNS 6 is - 0110 - so the ith - 2th means 3rd bit is 1
// cout << getIthBit(7, 1) << endl; // 1 - in BNS 7 is - 0111 - so the ith - 1st means 2nd bit is 1
// cout << getIthBit(5, 1) << endl; // 0 - in BNS 5 is - 0101 - so the ith - 1st means 2nd bit is 0
// }

// _____
```

#### 4) Set ith Bit - (sets the value to 1)





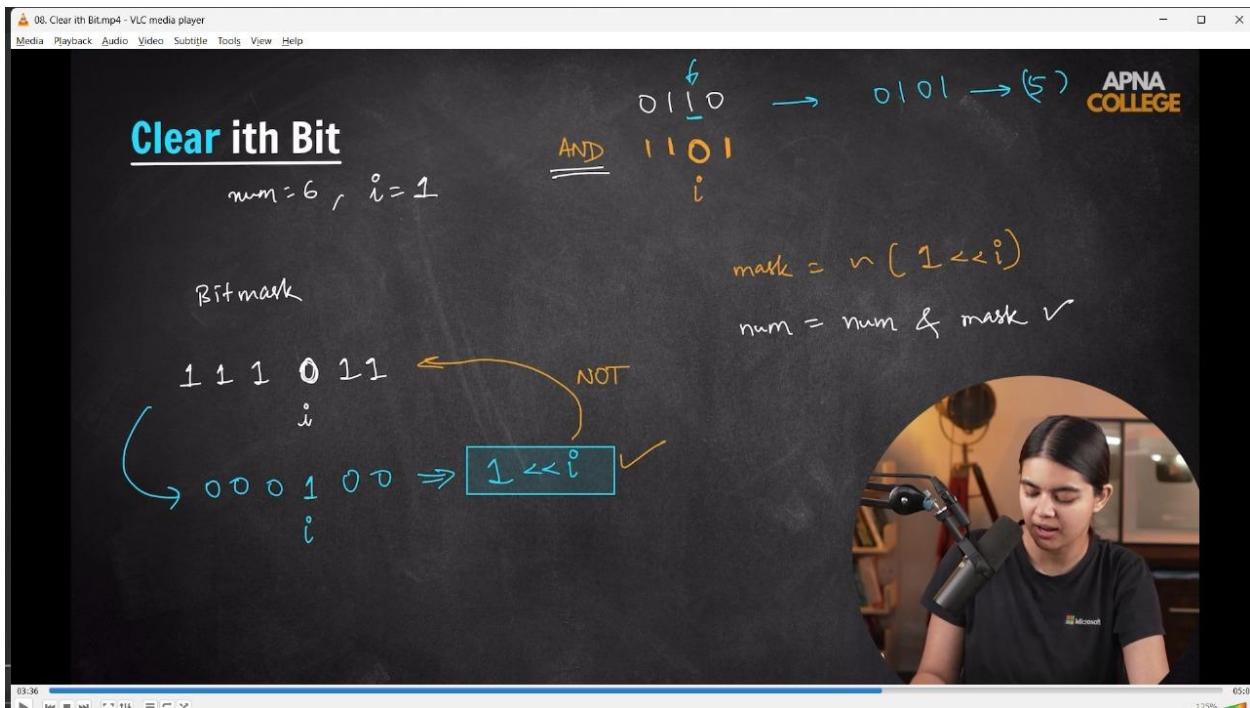
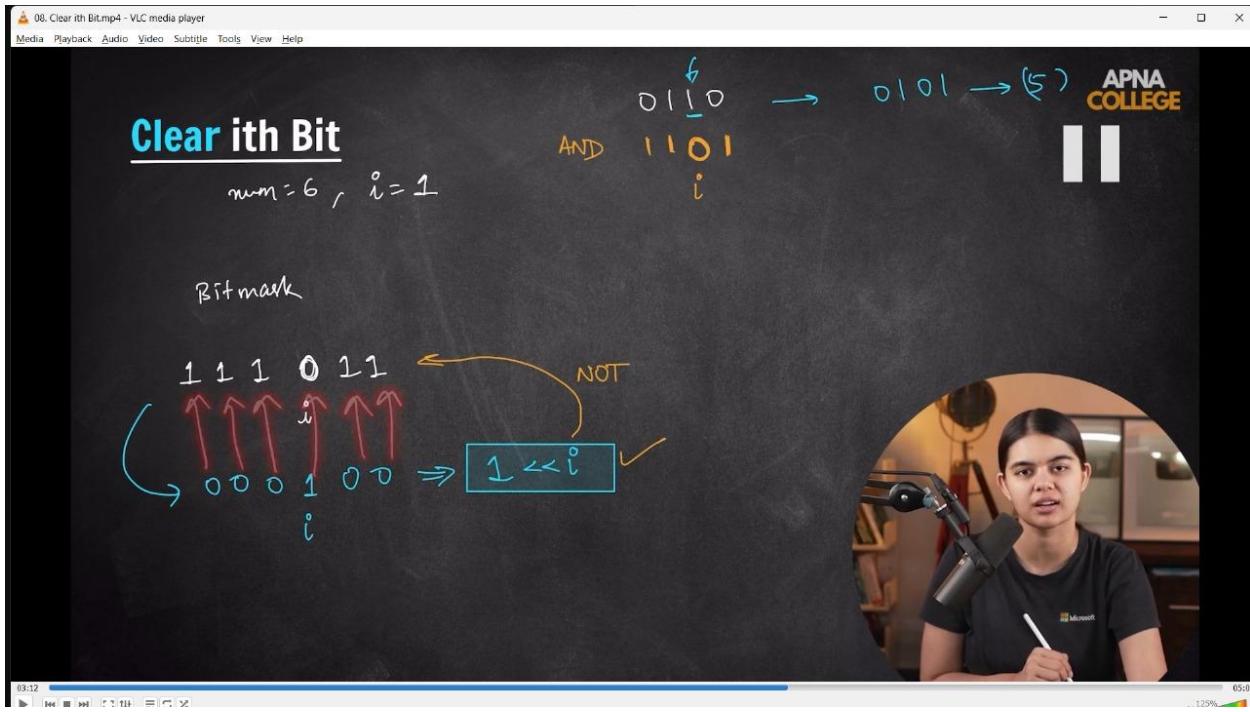
```

// int setIthBit(int num, int i)

// {
//     int BitMask = 1 << i;
//     return (num | BitMask);
// }

// int main()
// {
//     cout<<setIthBit(6,3)<<endl;//14 - in BNS 6 is - 0110 - so the ith - 3rd bit changes to 1 then it
// becomes 1110 which is 14
// }
// 
```

### 5) Clr ith Bit - (sets the value to 0)



```
// int ClearIthBit(int num, int i)
//{
//    int BitMask = ~(1 << i);
//    return num & BitMask;
```

```

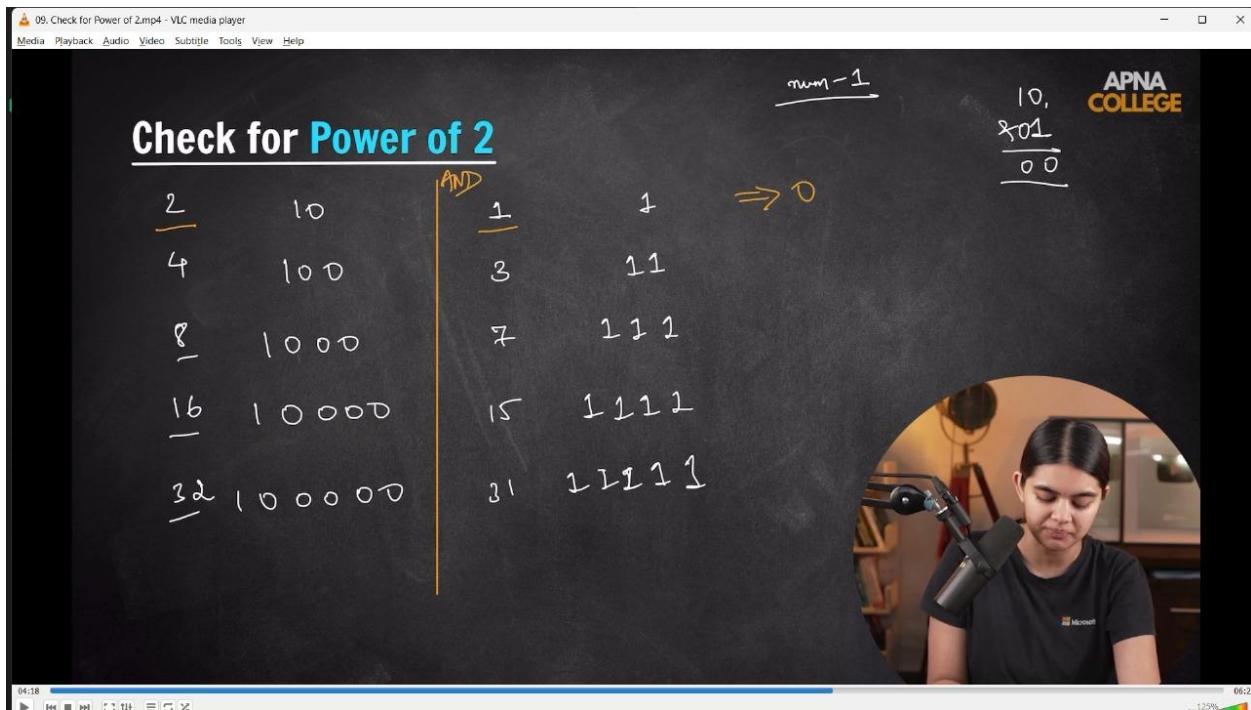
// }

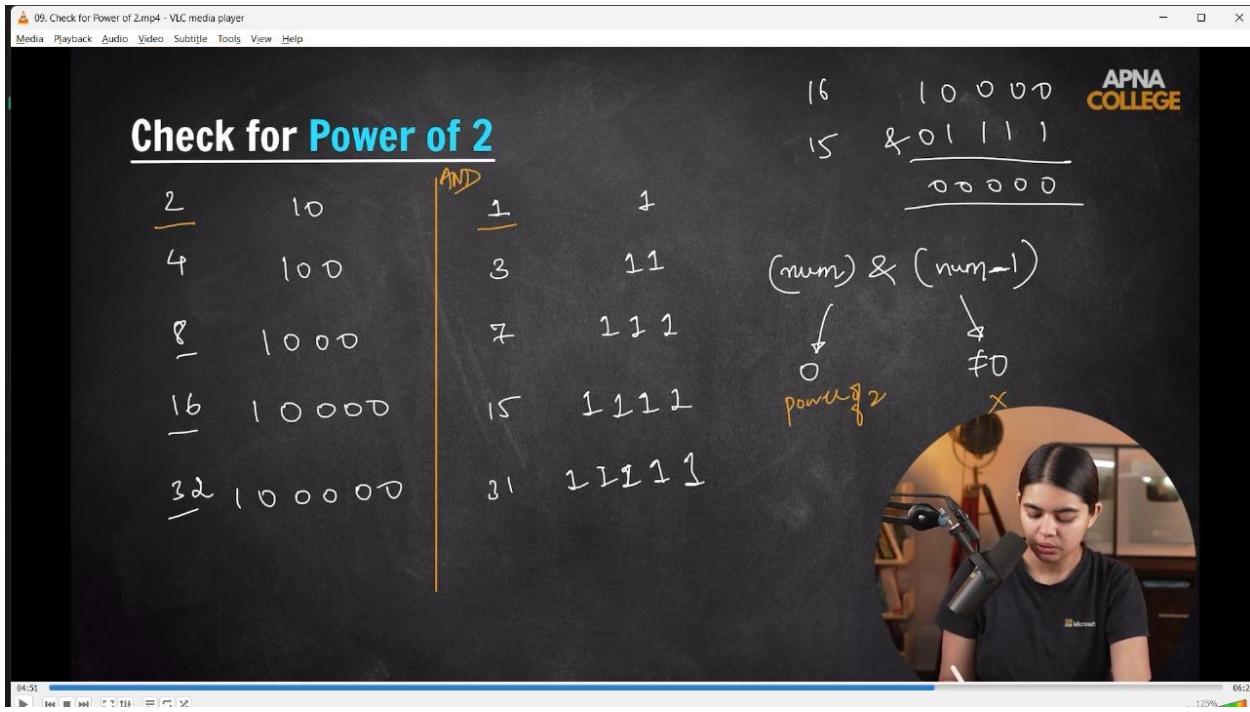
// int main()
// {
//   cout << ClearIthBit(6, 1) << endl;//4 - in BNS 6 is - 0110 - so the ith - 1st bit changes to 0 by
// using some op. then it becomes to 0100 which is 4

// }
// _____

```

## 6) No. is power of 2 -





```
/*
```

📌 Logic - Every no. which is power of two having only one 111 at LMSB, rest all the bits are 0 while in Odd no - the no. just above (-1) of power of 2 no. includes only 1 in the whole BN.  
So, the diff of it if ==0 then its an even no. otherwise odd no.

```
*/
```

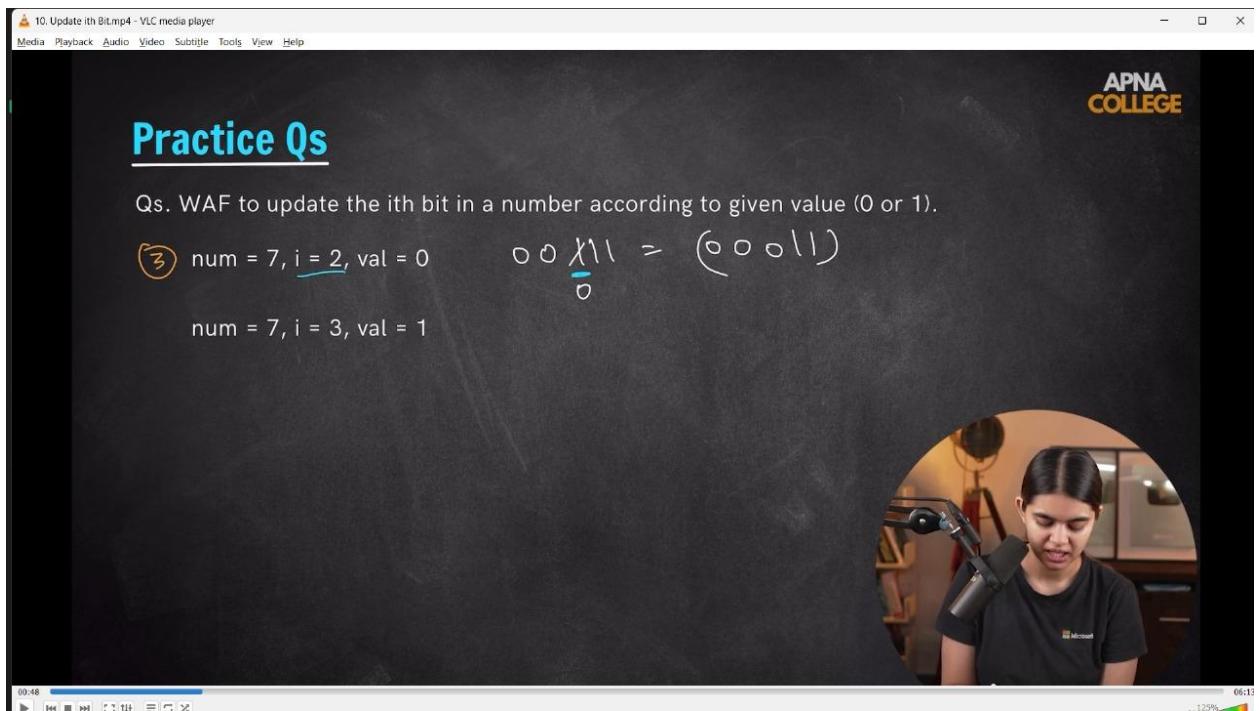
```
// bool ispowerof2(int num)
{
    // if (!(num & (num - 1)))
    {
        // return true;
    }
    // else
    {
        // return false;
    }
}
```

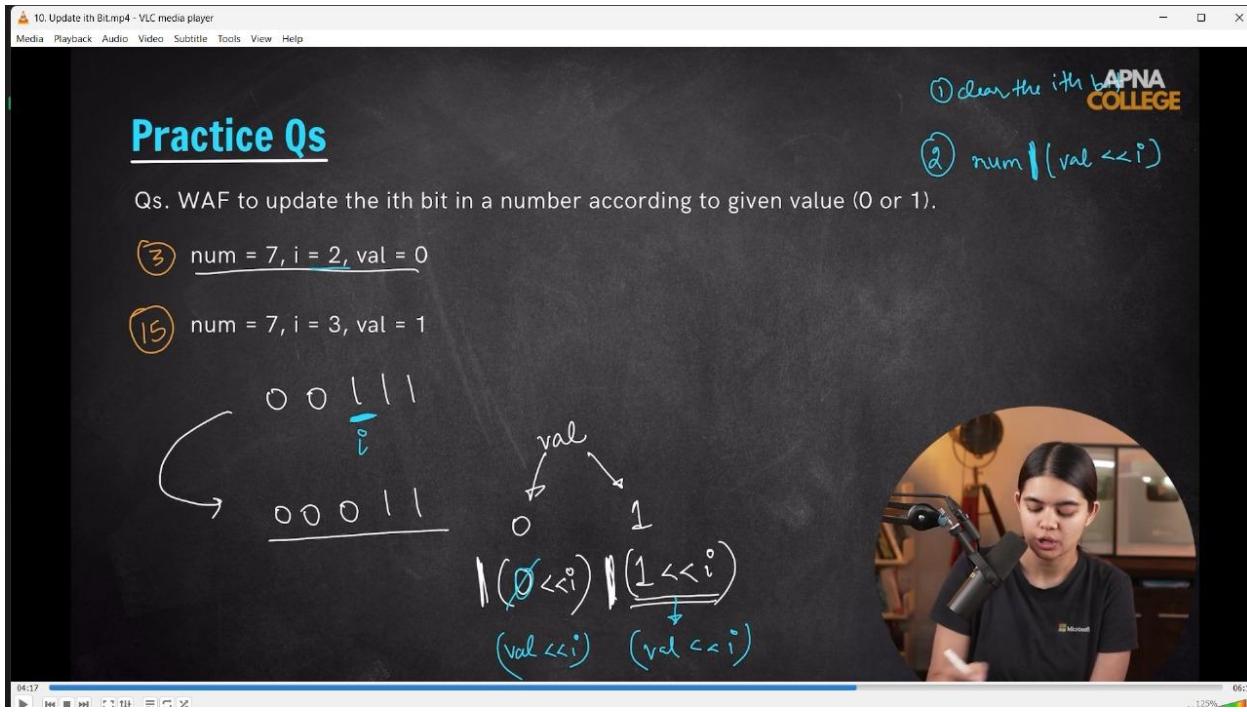
```

// }

// int main()
// {
//   cout<<ispowerof2(4)<<endl;//1
//   cout<<ispowerof2(31)<<endl;//0
// }
// _____
/// / QUn -
/// / 1) num = 7, i=2, val = 0
/// / 2) num = 7, i=3, val = 1;

```





```
// void updateIthBit(int num, int i, int val)

// {
//     num = num & ~(1 << i); // clear the ith bit
//     num = num | (val << i);
//     cout << num << endl;
// }

// int main()
// {
//     updateIthBit(7, 2, 0); // 3 - on num 7 whose value in BNS is 0111 and 2nd value in BNS
//     should be 0 - so its : 0011 which is = 3

//     updateIthBit(7, 3, 1); // 15 - on num 7 whose value in BNS is 0111 and 3rd value in BNS
//     should be 1 - so its : 1111 which is = 15

// }

// _____
/// // Qun - WAF to clear last i bits of a number - num = 15, i=2
```

11. Clear Last i Bits.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Practice Qs**

Qs. WAF to clear last  $i$  bits of a number.

num = 15,  $i = 2$

$00000111 \xrightarrow{\text{AND}}$

$\begin{array}{r} 00000111 \\ 111110000 \\ \hline 111110000 \end{array}$

$\sim 0 \ll i$

$0 \ll i$

$\sim 0 \ll i$

**APNA COLLEGE**

Volume 175%

num << 4

11. Clear Last i Bits.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Practice Qs**

Qs. WAF to clear last  $i$  bits of a number.

num = 15,  $i = 2$

$0 \rightarrow 00000000$

$\sim 0 \rightarrow 11111111$

$\sim 0 \ll i$

$0 \ll i$

**APNA COLLEGE**

num = num & ( $\sim 0 \ll i$ )

```
// void clearBits(int num, int i)

// {
//     int bitMask = (~0) << i;
//     num = num & bitMask;
```

```

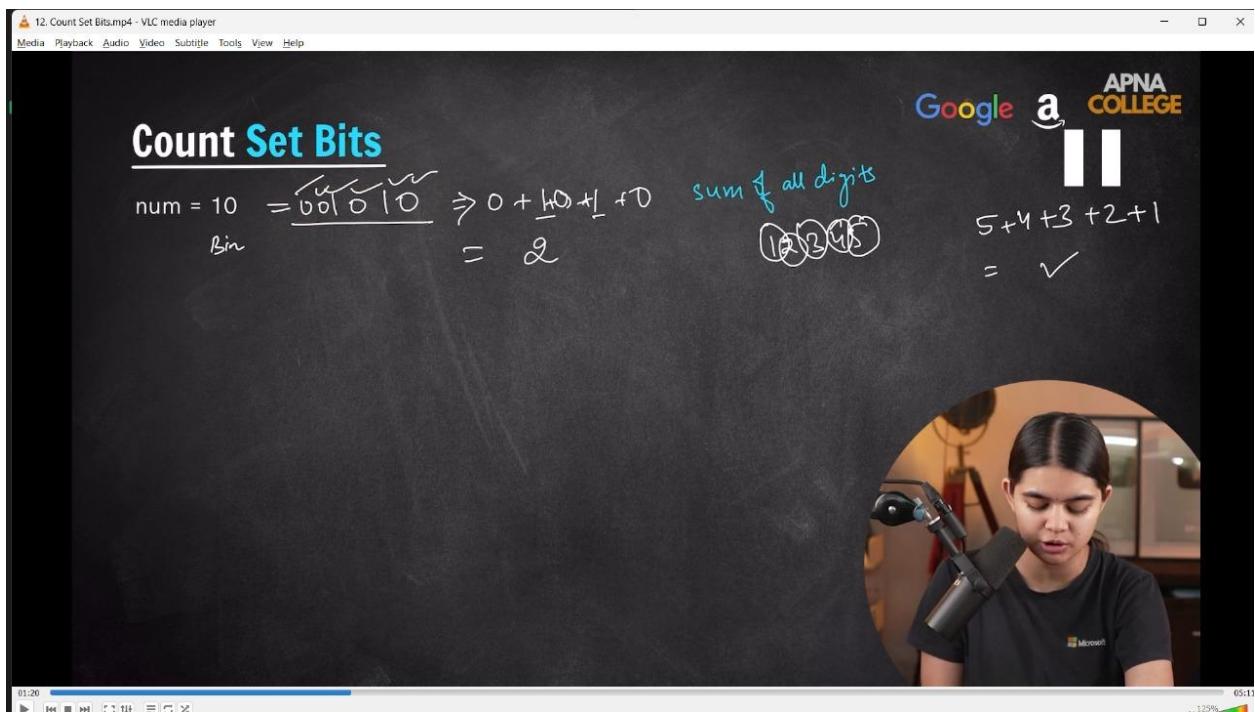
// cout << num << endl;
// }

// int main()
// {
//   clearBits(15, 2); // 12 - the num 15 in BNS is 1111, so 2bits clr from last then the no will be
//   1100 - which is equal to 12.

// }
// _____

```

### //7) Count of Set Bits -



12. Count Set Bits.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Count Set Bits

$\text{num} = 10$

$$\begin{array}{r} 10 \\ \times 2 \\ \hline 0 \\ \hline \end{array}$$

① last digit of binary form (odd/even)

$\text{num} \& 1$

$\begin{array}{c} \downarrow \\ 0 \end{array}$      $\begin{array}{c} \downarrow \\ 1 \end{array}$

(0)    (1)

1    0 ✓

12. Count Set Bits.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Count Set Bits

$\text{num} = 10$

$\text{num} > 0$

① last digit of binary form (odd/even)

$\text{num} \& 1$

$\begin{array}{c} \downarrow \\ 0 \end{array}$      $\begin{array}{c} \downarrow \\ 1 \end{array}$

(0)    (1)

②  $\text{num} >> 1$

$\text{count} = 0 + D + 1 + D + 1$

```
// int CountSetBits(int num)

// {
//     int count = 0;
//     while (num>0)
```

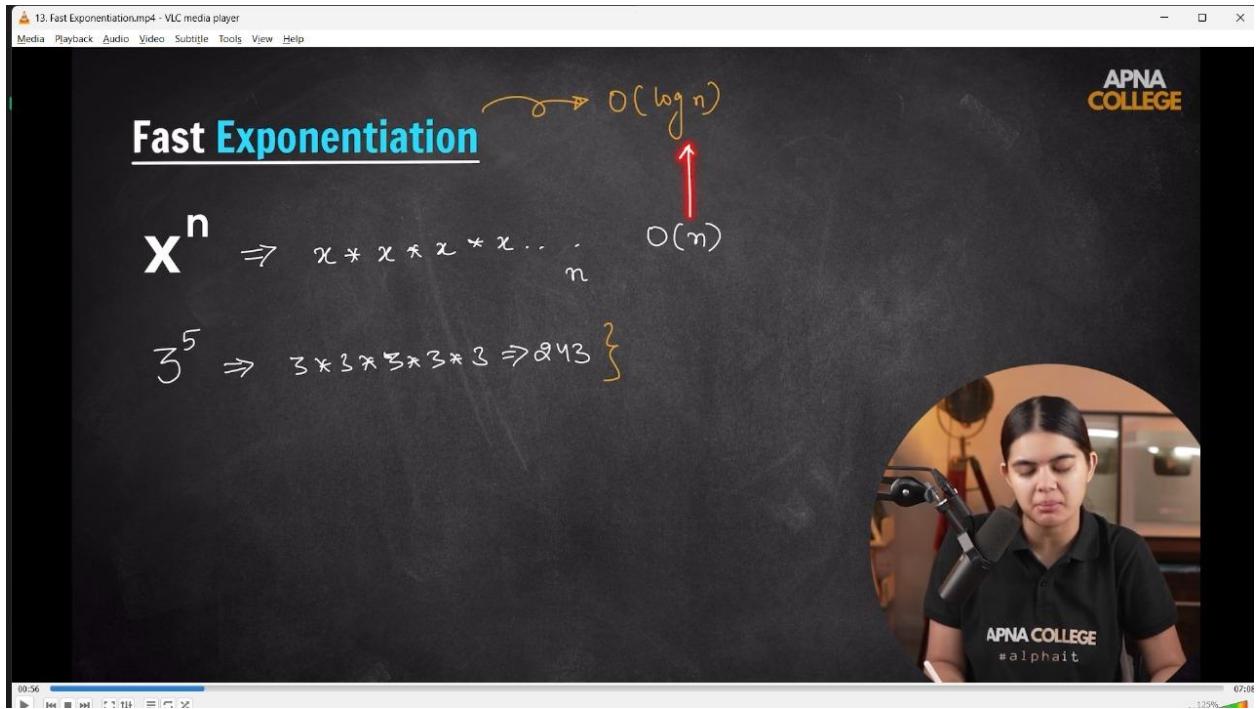
```

// {
//     int lastDig = num&1;
//     count+=lastDig;
//     num = num>>1;
// }
// cout<<count<<endl;
// return count;
// }

// int main()
//{
// CountSetBits(10); //2 - in BNS 10 is 001010, so its count is 0+0+1+0+1+0 = 2
//}
// _____

```

### //8) Fast Exponentiation -



13. Fast Exponentiation.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Fast Exponentiation

$x^n$

$n = 5$

$x = 3$

$ans = 1$

$(n) 5 \rightarrow 101 ; x = 3$

$\textcircled{1} \quad ans = ans * x = \textcircled{2}$   
 $x = x * x = 3^2 = 9$

$\textcircled{2} \quad x = x * x = 3^2 * 3^2 = 3^4 = \textcircled{3}$

$\textcircled{3} \quad ans = ans * x = 3 * 81 = \boxed{243}$

$5 \rightarrow \log_2 5 + 1 = 2 + 1 = 3$

$n \rightarrow \boxed{\log_2 n + 1} \approx \log_2 n$

$\rightarrow \underline{\underline{n}} \quad \underline{\underline{O(n)}}$

$\rightarrow \underline{\underline{\text{binary form of } Pow(n)}}$

$n \rightarrow \boxed{\log_2 n}$

APNA COLLEGE

14. Fast Exponentiation (Code).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Fast Exponentiation

$x^n$

$3^4$

$y \rightarrow 100 ; x = 3$

$ans = 1$

$\textcircled{1} \quad x = 3^2 = 9$

$\textcircled{2} \quad x = 3^2 * 3^2 = 81$

$\textcircled{3} \quad ans = ans * x = 1 * 81 = \boxed{81}$

APNA COLLEGE

```
// void fastExpo(int x, int n)
{
    int ans = 1;
    while (n > 0)
```

```
// {
//     int lastbit = n & 1;
//     if (lastbit)
//     {
//         ans = ans * x;
//     }
//     x = x * x;
//     n = n >> 1;
// }
// cout<<ans<<endl;
// }

// int main()
//{
//     fastExpo(3,5);//243
// }
```

---

---

## [8] OOP's in CPP –

1) Classes & Object basic Intro -

    1.1) Access Modifiers

    1.2) Getters & Setters -

2) Encapsulations -

    2.1) This Constrcutor

    2.2) Types of Cosntrcutrs -

    2.3) Copy Constructor Concept -

    2.4) Shallo & Deep Copy -

3) Destructor -

4) Inheritance -

    4.1) Mode of Inharitance -

    4.2) types of Inheritance -

        4.2.1) Single Inheritance -

        4.2.2) Multi-Level Inheritance -

        4.2.3) Multiple Inheritance -

        4.2.4) Hierarchical Inheroitance -

        4.2.5) Hybrid Inheroitance -

5) Polymorphism -

    5.1) Compiletetime Polymorphism -

        5.1.1) Function Overloading -

        5.1.2) Function Overloading -

    5.2)Polymorphism -

        5.2.1)Function Overriding -

        5.2.2)Virtual Functions –

6) Abstraction -

7) Static Keyword -

8) Friend Class & Friend Function –

## 8) OOP's in CPP –

//1) Classes & Object basic Intro - 01

//4) Inheritance - 25

//5) Polymorphism - 47

The diagram on the chalkboard illustrates the relationship between classes and objects. It shows 'Classes & Objects' at the top, which points down to 'entities in the real world'. Below that, it says 'group of these entities'. To the right, there are two columns: 'Properties' and 'functions'. Under 'Properties', there are examples like 'Phone', 'Computer', 'Book', and 'Car'. Under 'functions', there are examples like 'price', 'update()', 'name()', and 'getPercentage()'. A bracket groups 'Phone', 'Computer', and 'Book' under the heading 'APNA COLLEGE'. Another bracket groups 'Car' and 'Student' under 'APNA COLLEGE'. A third bracket groups 'getPercentage()', 'Teacher', and 'Dots' under 'APNA COLLEGE'. A circular video frame in the bottom right corner shows a person speaking.

The diagram on the chalkboard illustrates the relationship between classes and objects. It shows 'Classes & Objects' at the top, which points down to 'entities in the real world'. Below that, it says 'blueprint' and 'group of these entities'. To the right, there are two columns: 'Properties / attribute' and 'functions / member functions / methods'. Below 'Properties / attribute', there is a box labeled 'Student' containing 'String name', 'float cgpa', and 'getPercentage()'. Below 'functions / member functions / methods', there is a box labeled 'Class' and another labeled 'Blueprint', both with multiple arrows pointing to them from below. A circular video frame in the bottom right corner shows a person speaking.



```
// class User  
  
// {  
//     int id;  
//     string username;  
//     string password;  
//     string bio;  
//     string profilephoto;  
  
//     void deactivate()  
//     {  
//         cout << "Do you really want to delete your account" << endl;  
//     }  
  
//     void editBio(string newBio)
```

```
// {
//     bio = newBio;
//     cout << "Bio updated successfully" << endl;
// }

// void changeProfile(string newprofile)
// {
//     profilephoto = newprofile;
//     cout << "You got a new profile" << endl;
// }

//};

// int main()
// {
//     User first;
// }

// -----
// AccessModifiers
```

02. Access Modifiers.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Access Modifiers

→ keywords

- private      data & methods accessible inside class (*sensitive*)
- public        data & methods accessible to everyone → main ✓
- protected     data & methods accessible inside class & to its derived class

User  
↓  
forward (private)

Account  
↓  
debit()✓

03:23 03:57 125%

### //1.2) Getters & Setters -

03. Getters & Setters.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Getters & Setters

↳ special methods

prop: name, cgpa → direct access

00:34 02:50 125%

```
// class Student
```

```
// {
```

```
//   string name;
```

```
// float cgpa;

// public:
// void getPercentage()
// {
//     cout << (cgpa * 10) << endl;
// }

// // Setters - basicly uses when don;t have direct access to the Private elements of class but
// can be modified through setters
// void setName(string nameval)
// {
//     name = nameval;
// }

// void setCgpa(float cgpaval)
// {
//     cgpa = cgpaval;
// }

// // Getters - the value which we set through setter can be accessed through getters, as can;t
// access to the pvt class, so can be done by this Getter,Setters
// string getName()
// {
//     return name;
// }

// float getCgpa()
// {
//     return cgpa;
```

```

// }

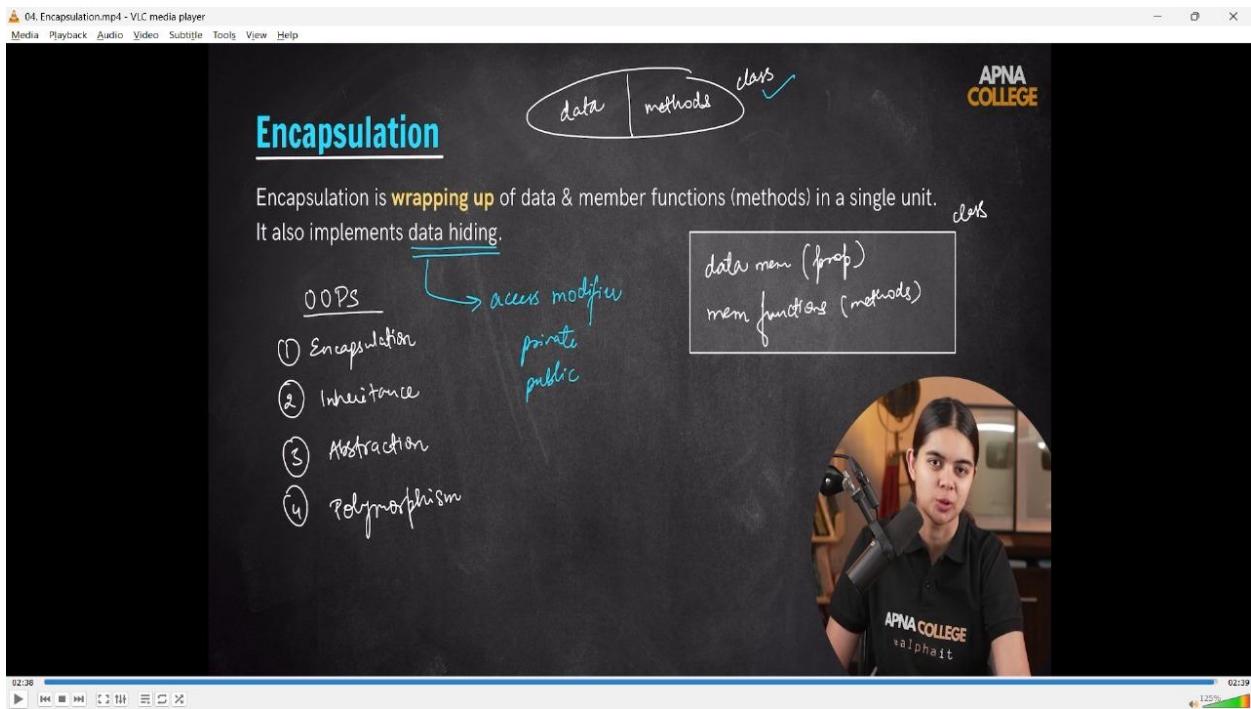
//};

// int main()
//{
// Student s1;
// s1.setName("Shubham");
// s1.setCgpa(7.55);

// cout<<s1.getName()<<endl;//Shubham
// cout<<s1.getCgpa()<<endl;//7.55
//}
// -----

```

## 2) Encapsulations -



//2) Encapsulations - EK single unit me data & member Functions ko implement krna. Matlab  
data member(property) & member function(methods)

// Abstraction - Constructor automatically create ho jaata h object creatin ke tym pr

```
// class Car
```

```
// {
```

```
//   string name;
```

```
//   string color;
```

```
// public:
```

```
//   Car(string nameValue, string colorvalue)
```

```
//   {
```

```
//     cout << "Constructor is called object being created---" << endl;
```

```
//     name = nameValue;
```

```
//     color = colorvalue;
```

```
//   }
```

```
// public:
```

```
//   void start()
```

```
//   {
```

```
//     cout << "Car has started.." << endl;
```

```
//   }
```

```
//   void stop()
```

```
//   {
```

```
//     cout << "Car has been stopped.." << endl;
```

```
//   }
```

```
// // Getter
// string getName()
// {
//     return name;
// }
//};

// int main()
//{
//    // Car c1;/// Constructor is called object being created----
//    Car c1("Airtroz", "Black");
//    cout << "Car name : " << c1.getName() << endl; // Car name : Airtroz
//}
// -----
//2.1) This Constrcutor
```

05.Constructor.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Constructor

Special method invoked automatically at time of **object creation**. Used for Initialisation.

- Same name as class
- Constructor doesn't have a return type
- Only called once (automatically), at object creation
- Memory allocation happens when constructor is called



✓ Car → name  
color ] start  
stop APNA  
COLLEGE

01:29 14:44 125%

05.Constructor.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Constructor

**this** is a special pointer in C++ that points to the current object.

*this->prop is same as \*(this).prop*



C1. name  
C1. name  
\* this. name

08:56 14:44 125%

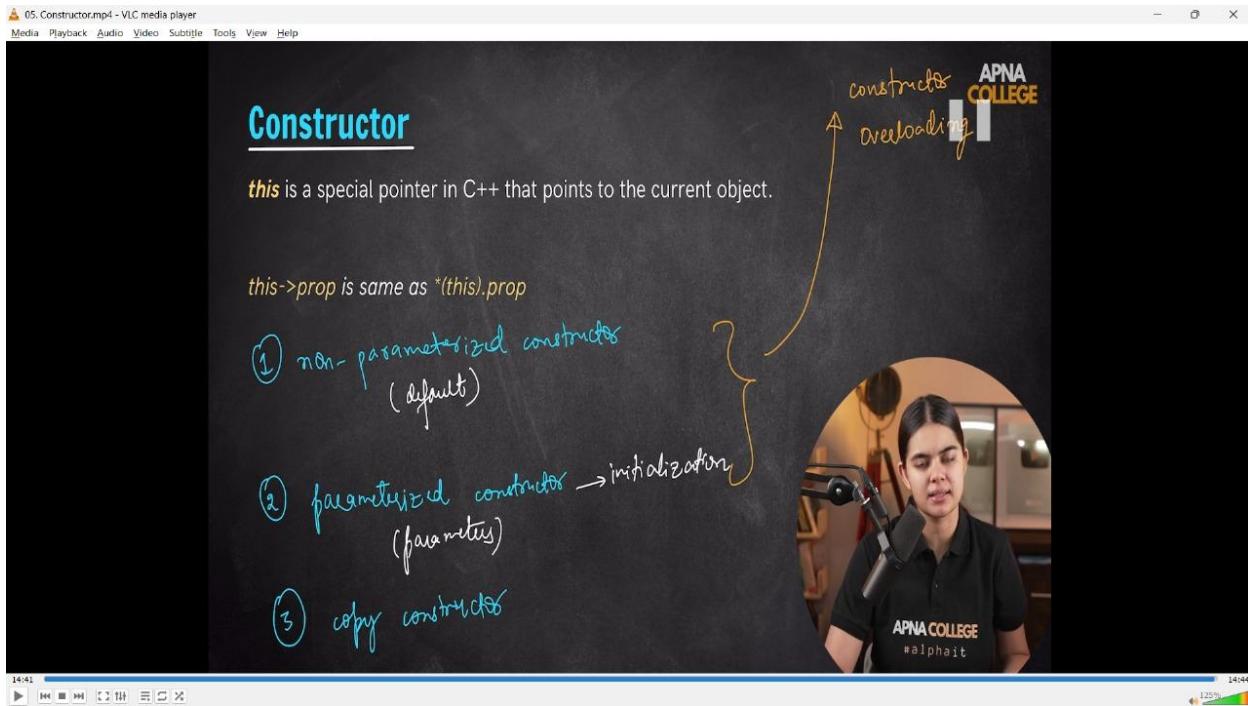
```
// This Constructor -this is a special pointer in c++ that points to the current object
// class Car1
//{
//    string name;
```

```
// string color;

// public:
// Car1(string name, string color)
// {
//     cout << "Constructor is called. Object being created ..." << endl;
//     this->name = name;// phle waala name jo object bnaya uska h & dusra name jo fun me
// pass kiya uska he
//     this->color = color;
// }
//};

// int main()
// {
//     Car1 c1("Airtroz","Black");
// }

// -----
//2.3) Types of Constructors -
```



```
// class Car2
//{
//    string name;
//    string color;

// public:
//    Car2() // Non- Parameterized Constructor
//    {
//        cout << "Constructore w/o parameters " << endl;
//    }
//    Car2(string name, string color) // Parameterized constructor
//    {
//        cout << "Constructor with Parameters - " << endl;
//        this->name = name;
//        this->color = color;
```

```
// }

// // Getter
// string getName()
// {
//     return name;
// }
//};

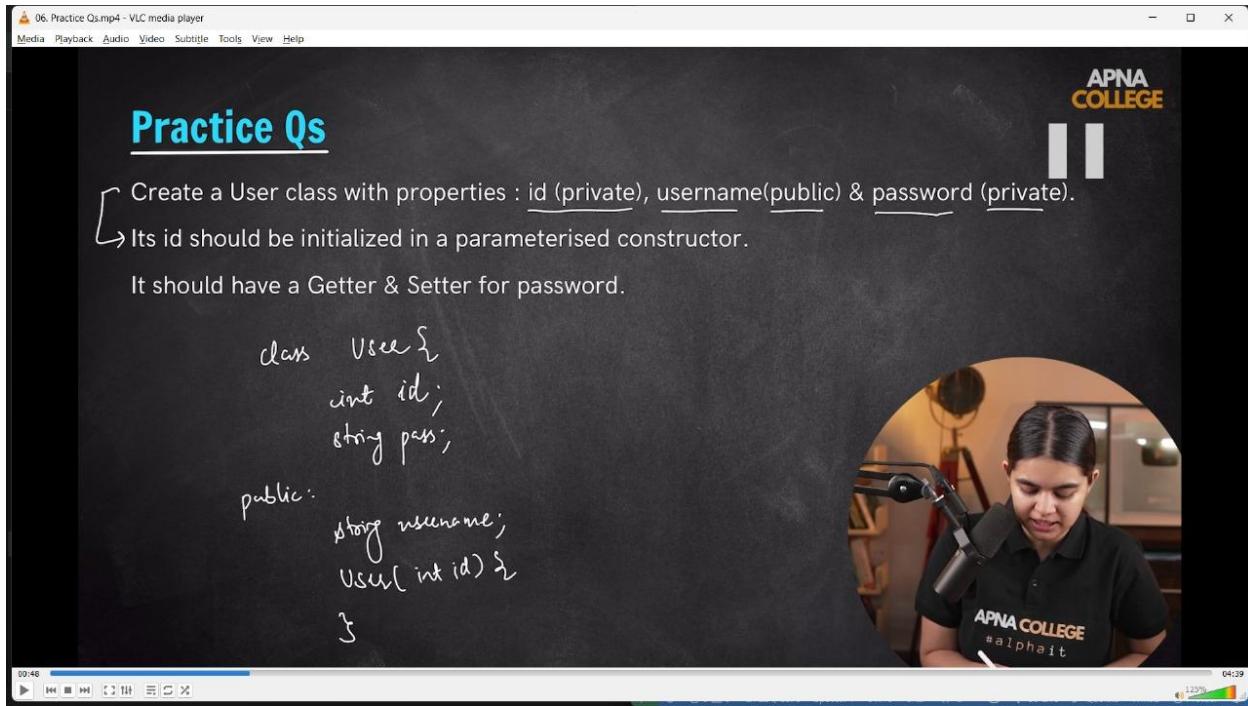
// int main()

// { Car2 c0;//anyh object created for the class. this is non araterized constrcuot
//   Car2 c1("Avtroz", "Black");
//   Car2 c2("Mahindra SUV AMT","Dark Blue");

// /*
// Constructore w/o parameters
// Constructor with Parameters -
// Constructor with Parameters -
// */
// }

// -----
```

### Practice Qs



→ Create a User class with properties:

id (private), username (public) & password (private).

Its id should be initialized in a parameterized constructor.

It should have a Getter & Setter for password.

```
/*
// class User
//{
// private:
//     int id;
//     string password;

// public:
//     string username;
```

```
// User(int id)
// {
//     this->id = id; // this mean jo bhi current id h isme created parameter waali id ki value aa
// kr store ho jaaygi
// }

// // Getter
// string getPassword()
// {
//     return password;
// }

// // Setter
// void setPassword(string password)
// {
//     this->password = password;
// }
// };

// int main()
// {
//     User user1(101);
//     user1.username = "MicrosftHyderabad";
//     user1.setPassword("Shubh@51LPA");

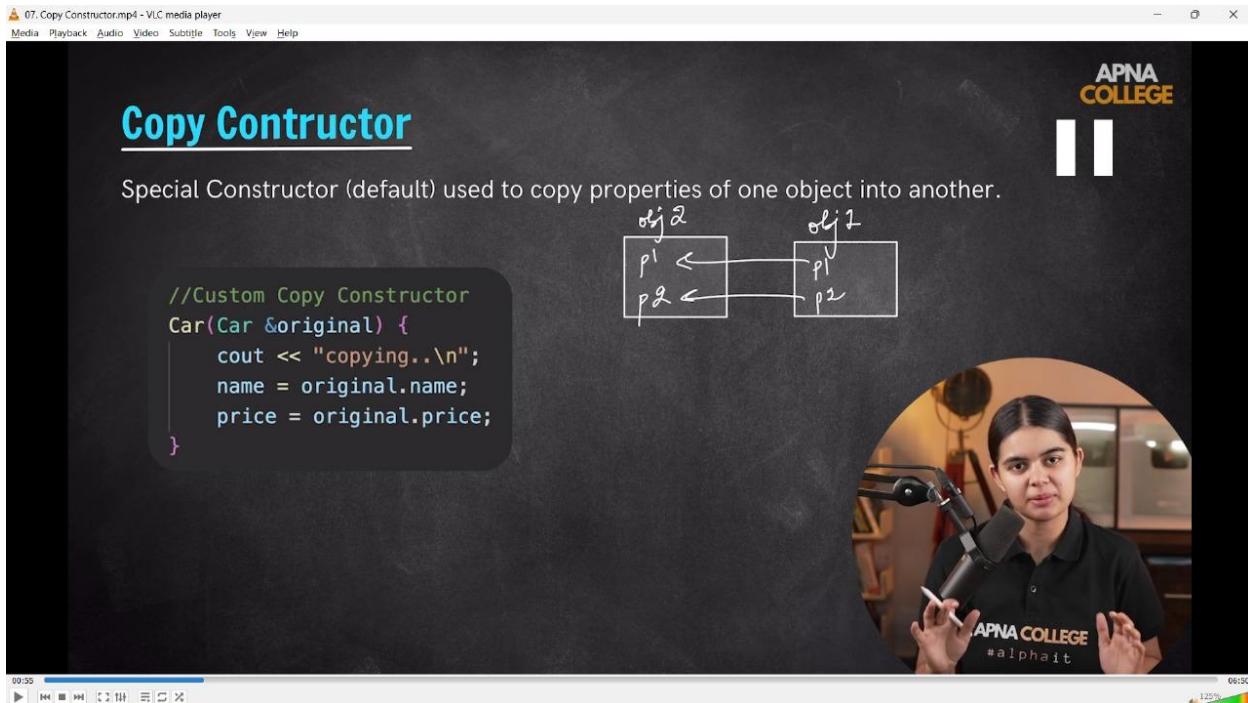
//     cout << "username : " << user1.username << endl;    // username : MicrosftHyderabad
```

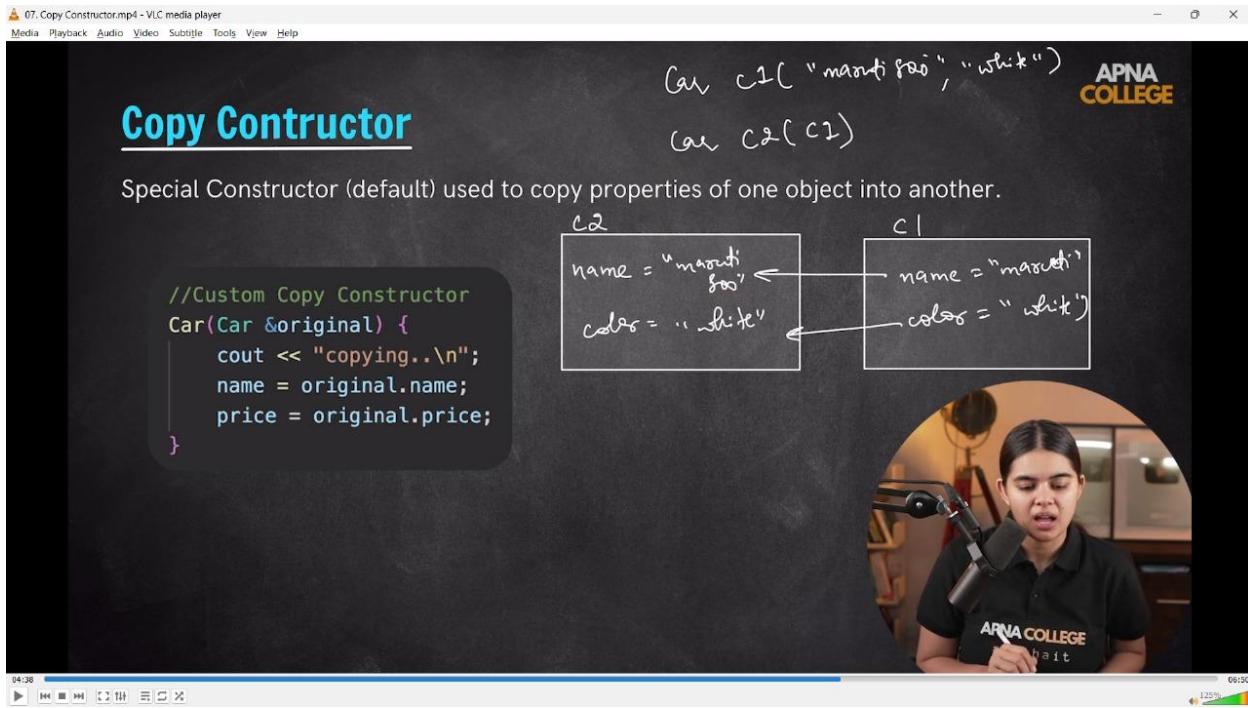
```
// cout << "password : " << user1.getPassword() << endl; // password : Shubh@51LPA
```

```
// }
```

```
// -----
```

## //2.4) Copy Constructor Concept -





```
// class Car
//{
// public:
//     string name;
//     string color;

//     Car(string name, string color) // Parameterized constructor
//     {
//         this->name = name;
//         this->color = color;
//     }

//     Car(Car &original) // creating a copy constructor from the given original constructor at the
//     same time
//     {

```

```

//      cout << "Copying original to new...." << endl;
//      name = original.name;
//      color = original.color;
//  }
//};

// int main()
//{
//  Car c1("Maninda SUV 700", "Dark Blue");

//  Car c2(c1);
//  cout << c2.name << endl; // Maninda SUV 700
//  cout << c2.color << endl; // Dark Blue

//  // - by using the c1 we could access through c2. at where we passed c1/ So w/o
//  creating a seperate cosntruuctor using c1 we could acess.

//  // Own custom copy constrcutor -
//  Car c3(c1);//Copying original to new....
//  cout << c3.name << endl;//Maninda SUV 700
//  cout << c3.color << endl;//Dark Blue

// }

// -----
//2.5) Shallow & Deep Copy -

```

06. Shallow & Deep Copy.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Shallow & Deep Copy

Shallow copy copies references to original array. But array remains same.  
Deep copy created a brand new copy of the array.

Compiler generally creates a shallow copy for array.

We need to define own copy constructor when deep copy is needed i.e. when class contains pointers to dynamically allocated memory.

C2

C1

name = ""

color = ""

int\* mileage = 100

\* mileage

int mileage = 100

APNA COLLEGE #alphaIt

02:40 12:21

06. Shallow & Deep Copy.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Shallow & Deep Copy

Shallow copy copies references to original array. But array remains same.  
Deep copy created a brand new copy of the array.

Compiler generally creates a shallow copy for array.

We need to define own copy constructor when deep copy is needed i.e. when class contains pointers to dynamically allocated memory.

C2

C1

name = ""

color = ""

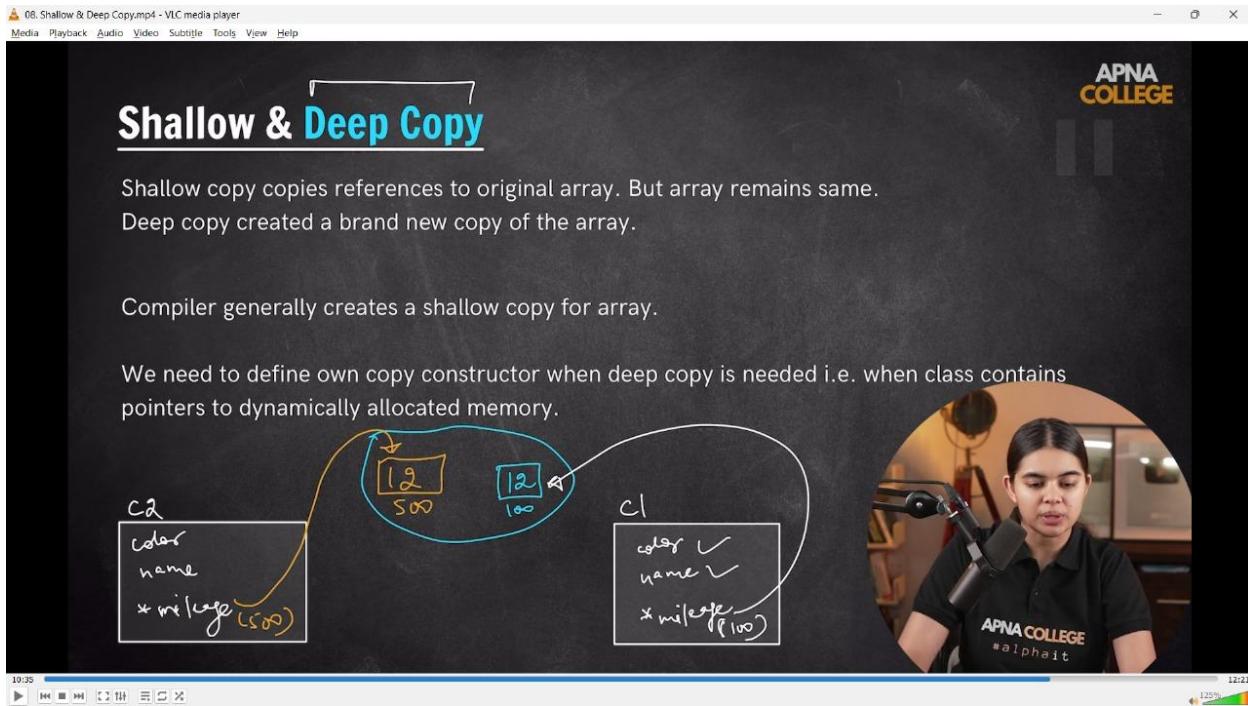
int\* mileage = 100

\* mileage

int mileage = 100

APNA COLLEGE #alphaIt

07:00 12:21



/\*

-copy constr. concept me saare static var to copy ho jaate h as it is from original memory

but dynamic constr. not copy while they originally copy from the heap memo and even on changing in copies constrcutor the changes reflected in original constrctr adn heap emo. too

\*/

// class Car

// {

// public:

// string name;

// string color;

// int \*mileage;

// Car(string name, string color) // Parameteized constructor

// {

```
//      this->name = name;
//      this->color = color;// These 2 are static allocation with proper copy from main meo
//      mileage = new int;// Dynamic Allocation - copy from heap memo on the behalf of address
& directly change in main and hap memo
//      *mileage = 12;
//  }

//  Car(Car &original) // creating a copy constructor from the given original constructor at the
same time

//  {
//      cout << "Copying original to new...." << endl;
//      name = original.name;
//      color = original.color;
//      mileage = original.mileage;
//  }
//};

// int main()
// {
//     Car c1("Maninda SUV 700", "Dark Blue");

// // Own custom copy constructor -
//     Car c3(c1);//Copying original to new....
//     cout << c3.name << endl;//Maninda SUV 700
//     cout << c3.color << endl;//Dark Blue
//     cout << *c3.mileage << endl;//12
//     *c3.mileage = 10;
```

```

// cout<<*c1.mileage<<endl;//10 - so we did the exchange in c3 but also goit the change in c1
as in pointer or address also changes in main addres

// }

// -----
// what if i do changes in Original construcotr not the custom created constructor-
// class Car

//{
// public:
//   string name;
//   string color;
//   int *mileage;

//   Car(string name, string color) // Parameteized constructor
//   {
//     this->name = name;
//     this->color = color; // These 2 are static allocation with proper copy from main memo
//     mileage = new int; // Dynamic Allocation - copy from heap memo on the behalf of
//     address & directly change in main and heap memo
//     *mileage = 12;
//   }
//};

// int main()
//{
//   Car c1("Maninda SUV 700", "Dark Blue");
//   // On changing in the main original constr.
//   Car c2(c1);
//   cout << c2.name << endl; // Maninda SUV 700

```

```

// cout << c2.color << endl; // Dark Blue
// cout << *c2.mileage << endl; // 12
// *c2.mileage = 10;
// cout << *c1.mileage << endl; // 10 - so we did the exchange in c2 which is in original constructor
but also goit the change in c1 as in pointer or address also changes in main address
// }

// -----

```

### //3) Destructor -



// Destructor - It deletes the created constructors automatically, but for custom constructors need to create destructor manually

```

// class Car
//{
// public:
//     string name;
//     string color;
//     int *mileage;

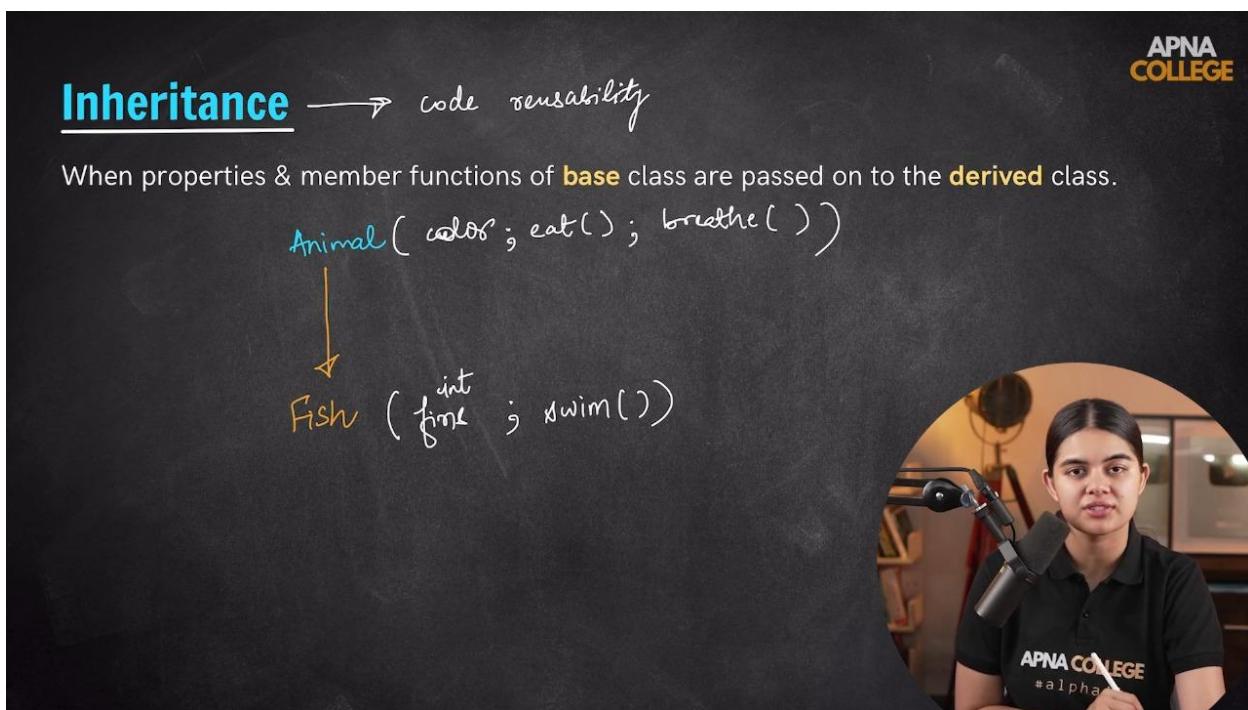
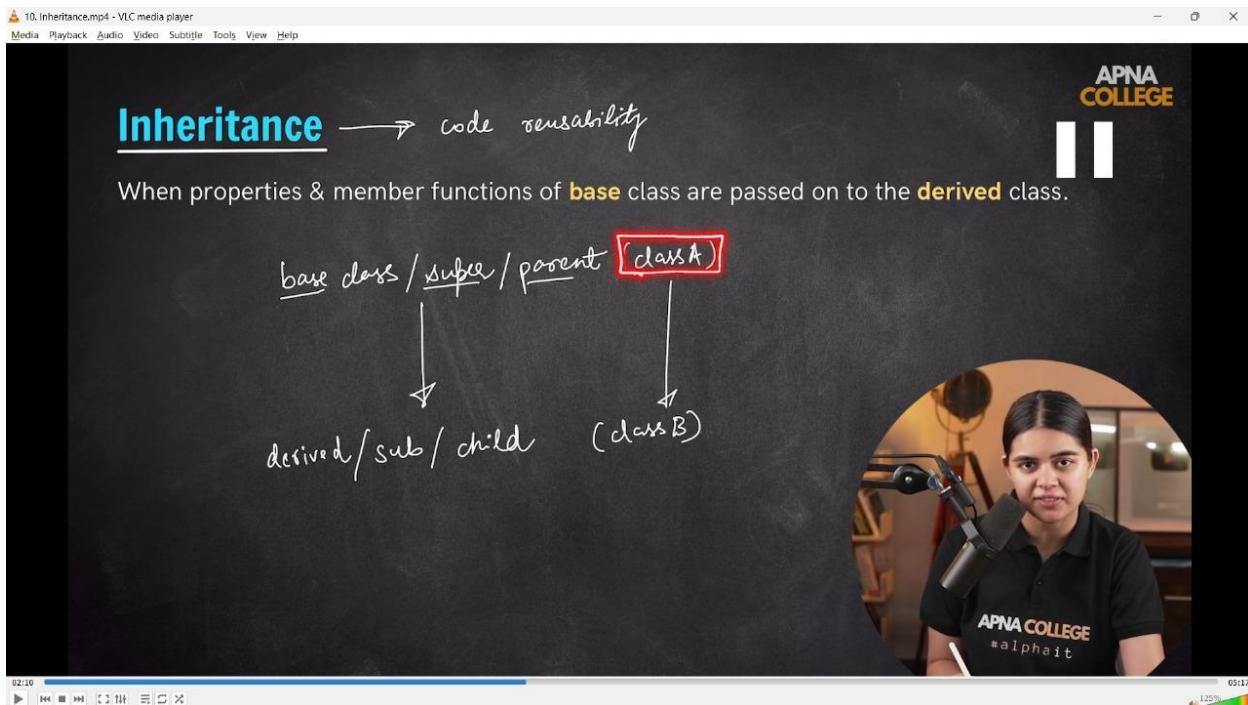
```

```
// Car(string name, string color) // Parameteized constructor
// {
//     this->name = name;
//     this->color = color; // These 2 are static allocation with proper copy from main memo
//     mileage = new int; // Dynamic Allocation - copy from heap memo on the behalf of
//     address & directly change in main and heap memo
//     *mileage = 12;
// }
// ~Car()
// {
//     cout << "Deleting Object - " << endl;
// }
// };
// int main()
// {
//     Car c1("Maninda SUV 700", "Dark Blue");
//     cout << c1.name << endl;
//     cout << c1.color << endl;
//     cout << *c1.mileage << endl;
//     /*
//     Dark Blue
//     12
//     Deleting Object -
//
//     */
```

// }

// \_\_\_\_\_

## 4) Inheritance –



### // 4.1) Mode of Inheritance –

Case I

Case II

### Case III

11. Mode of Inheritance.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

The diagram illustrates the modes of inheritance:

- Mode of Inheritance:** private, public, protected.
- Type of Inheritance:** Public, Protected, Private.

Base class member access specifier	Type of Inheritance		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

Annotations on the chalkboard include:
 

- Red circles highlight 'Public' in the first two rows.
- A red X is placed over the 'Not accessible (Hidden)' row.
- A red arrow points from 'Public' in the first row to 'Protected' in the second row.
- A red arrow points from 'Protected' in the first row to 'Protected' in the second row.
- A red arrow points from 'Private' in the first row to the 'X' in the second row.

Chalkboard annotations on the right side:
 

- Base/Parent (Animal)
- APNA COLLEGE
- ↓
- public
- child (Fish)

Video player controls: 00:06 / 07:30

11. Mode of Inheritance.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

The diagram illustrates the modes of inheritance:

- Mode of Inheritance:** private, public, protected.
- Type of Inheritance:** Public, Protected, Private.

Base class member access specifier	Type of Inheritance		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

Annotations on the chalkboard include:
 

- Red circles highlight 'Public' in the first two rows.
- A red X is placed over the 'Not accessible (Hidden)' row.
- A red arrow points from 'Public' in the first row to 'Protected' in the second row.
- A red arrow points from 'Protected' in the first row to 'Protected' in the second row.
- A red arrow points from 'Private' in the first row to the 'X' in the second row.

Chalkboard annotations on the right side:
 

- private ✓
- ↳ child
- inside class ✓
- + derived class

Video player controls: 04:00 / 07:30

```
// Case - 1 - when both(parent & child class) are public
```

```
// class Animal
```

```
// {
```

```
// public:  
//   string color;  
  
//   void eat()  
//   {  
//     cout << "eats -" << endl;  
//   }  
  
//   void breathe()  
//   {  
//     cout << "breathes - " << endl;  
//   }  
// };  
  
// class Fish : public Animal // Inheritance.  
// {  
// public:  
//   int fins;  
  
//   void swim()  
//   {  
//     cout << "swims - " << endl;  
//   }  
// };  
// /*  
//  - In inheritance here is 2 classes already defined eat and breathe,
```

```
// so the 3rd class forfish which also having these properties and one more property of swim  
which  
  
// can be declare in this class and can be inherited from other class too.  
  
// */  
  
// int main()  
  
// {  
  
//     Fish f1;  
  
//     f1.fins = 3;  
  
//     cout << f1.fins << endl;  
  
//     f1.swim(); // swims  
  
//     f1.eat(); // eats  
  
//     f1.breathe(); // breathes  
  
//     /* So, how this 3rd class fish is able to access the elements of above 2 classes, by using  
Inheritance  
  
//     */  
  
// }  
  
// -----
```

```
// What if I do the same for Protected class -  
  
// Case - II - when parent class is public but the child class is protected - cannot be accessed by  
main fun  
  
// class Animal  
  
// {  
  
// public:  
  
//     string color;  
  
  
//     void eat()
```

```
// {
//     cout << "eats -" << endl;
// }

// void breathe()
// {
//     cout << "breathes - " << endl;
// }
//};

// class Fish2 : protected Animal // Inheritance.

//{
// public:
//     int fins2;

//     void swim2()
//     {
//         cout << "swims - " << endl;
//     }
// };

// int main()
// {
//     Fish2 f1;
//     f1.fins2 = 3;
//     cout << f1.fins2 << endl;
```

```
// f1.swim2(); // swims

/* f1.eat(); // error: 'void Animal::breathe()' is inaccessible within this context

// f1.breathe(); // error: 'Animal' is not an accessible base of 'Fish2'

// SO clearly inaccesible from main fun if class is protected , but can be accessed within the same
class

*/
// }

// -----
// But within the calss, it can be accessible - inside the protected child class public parent class
can be accessed, not from main fun.
```

```
// class Animal

// {

// public:

//     string color;

//     void eat()

//     {

//         cout << "eats -" << endl;

//     }

//     void breathe()

//     {

//         cout << "breathes - " << endl;

//     }

// };
```

```
// class Fish2 : protected Animal // Inheritance.  
// {  
// public:  
//   int fins2;  
  
//   void swim2()  
//   {  
//     eat();  
//     cout << "swims - " << endl;  
//   }  
// };  
  
// int main()  
// {  
//   Fish2 f1;  
//   f1.fins2 = 3;  
//   cout << f1.fins2 << endl; // 3  
//   f1.swim2();  
//   /*  
//    eats  
//    swims  
//    So, when the inside the protrected class another classes are defined then can be accessed  
//    from main func  
//    */  
// }  
// -----
```

```
// Case III - when the the parent class is by default Private & the child class is Public -
```

```
// class Animal
```

```
// {
```

```
// private:
```

```
//   string color;
```

```
//   void eat()
```

```
//   {
```

```
//     cout << "eats -" << endl;
```

```
// }
```

```
//   void breathe()
```

```
//   {
```

```
//     cout << "breathes - " << endl;
```

```
// }
```

```
//};
```

```
// class Fish2 : public Animal // Inheritance.
```

```
// {
```

```
// public:
```

```
//   int fins2;
```

```
//   void swim2()
```

```
//   {
```

```
//     // eat();
```

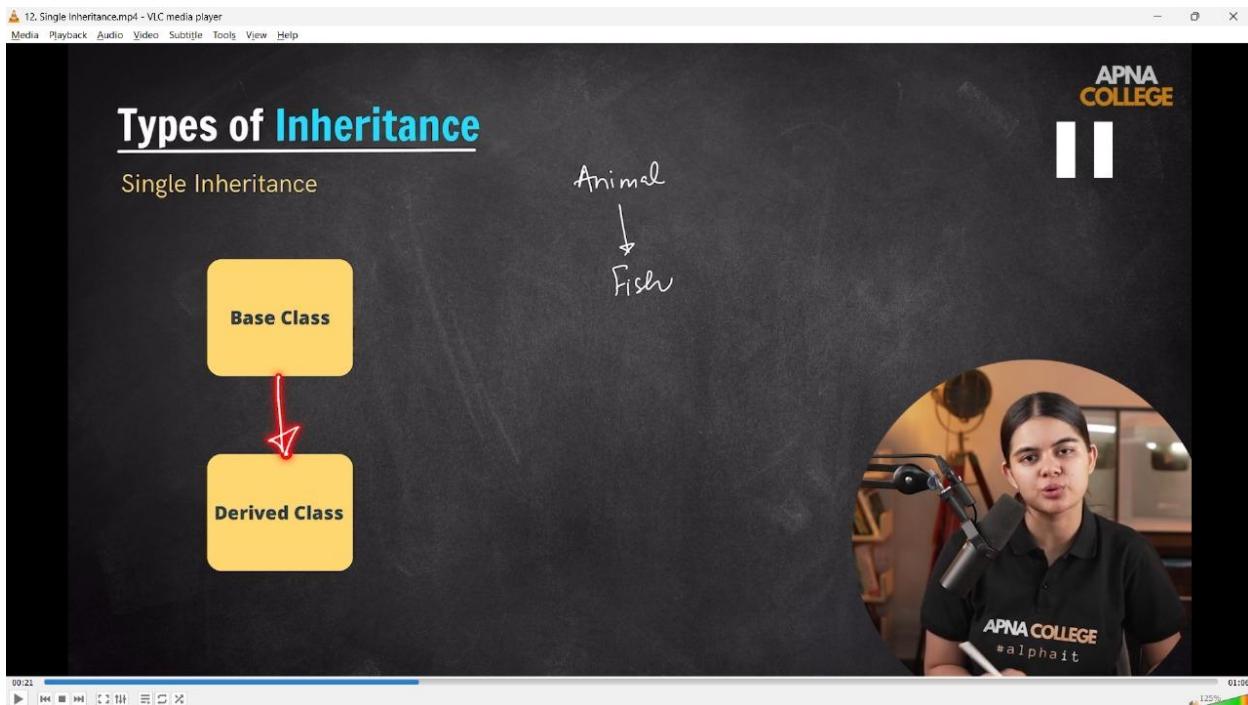
```
//     cout << "swims - " << endl;
//  }
//};

// int main()
//{
//    Fish2 f1;
//    f1.fins2 = 3;
//    cout << f1.fins2 << endl; // 3
//    f1.swim2();
// /* O/p -
// Inheritance.cpp: In member function 'void Fish2::swim2()':
// Inheritance.cpp:167:9: error: 'void Animal::eat()' is private within this context
//     eat();
//     ^
// Inheritance.cpp:149:10: note: declared private here
// void eat()
//     ^
// Inheritance.cpp:167:13: error: 'void Animal::eat()' is private within this context
//     eat();
//     ^
// Inheritance.cpp:149:10: note: declared private here
// void eat()
//     ^
// So, in the case where Parent class is Pvt then not access to the child class either it's
// public or protected
```

```
// */  
// }  
// _____
```

## //4.2) types of Inheritance -

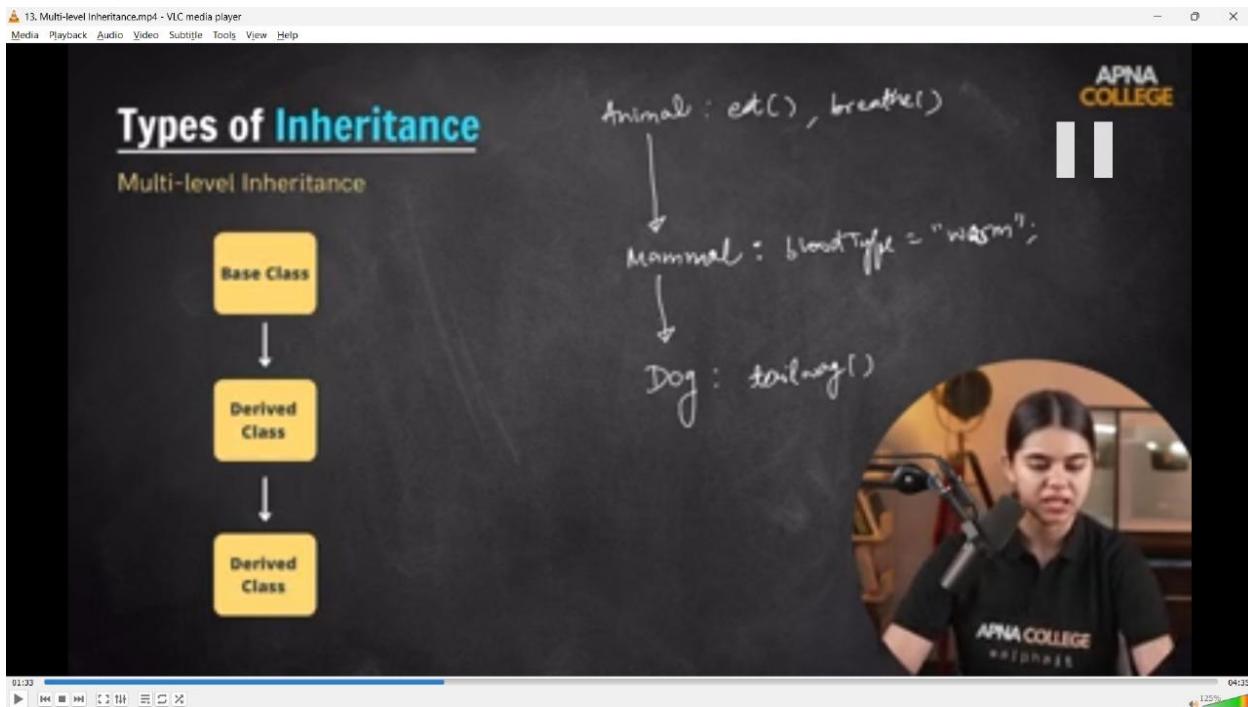
### //4.2.1) Single Inheritance -



```
// class Animal  
// {  
// public:  
//     string color;  
  
//     void eat()  
//     {  
//         cout << "eats -" << endl;  
//     }
```

```
// void breathe()  
// {  
//     cout << "breathes - " << endl;  
// }  
// };  
  
// class Fish : public Animal // Inheritance.  
// {  
// public:  
//     int fins;  
  
//     void swim()  
//     {  
//         cout << "swims - " << endl;  
//     }  
// };  
  
// int main()  
// {  
//     Fish f1;  
  
//     f1.swim(); // swims  
//     f1.eat(); // eats  
//     f1.breathe(); // breathes  
// }
```

## //4.2.2) Multi-Level Inheritance -



```
// class Animal
//{
// public:
//     void eats()
//     {
//         cout << "Eats..." << endl;
//     }
//     void breath()
//     {
//         cout << "Breaths..." << endl;
//     }
//};
```

```
// class Mammal : public Animal // Mammal class inherits all properties of Animal Class
```

```
// {  
// public:  
//     string bloodtype = "Warm";  
//     // Or can also be called byh constructor -  
//     string bloodcolor;  
//     Mammal()  
//     {  
//         bloodcolor = "Greenish";  
//     }  
// };
```

```
// class Dog : public Mammal// while the dog class inherits all properties of Mammal class as  
well as Animal class too
```

```
// {  
// public:  
//     void tailmovement()  
//     {  
//         cout << "Dog can also wipe out its tail everytim,e" << endl;  
//     }  
// };
```

```
// int main()  
// {  
//     Dog d1;  
//     d1.eats();  
//     d1.breath();
```

```

// d1.tailmovement();

// cout << "So, the colour of blood is - " << d1.bloodcolor << endl;

// /*
// Eats...

// Breaths...

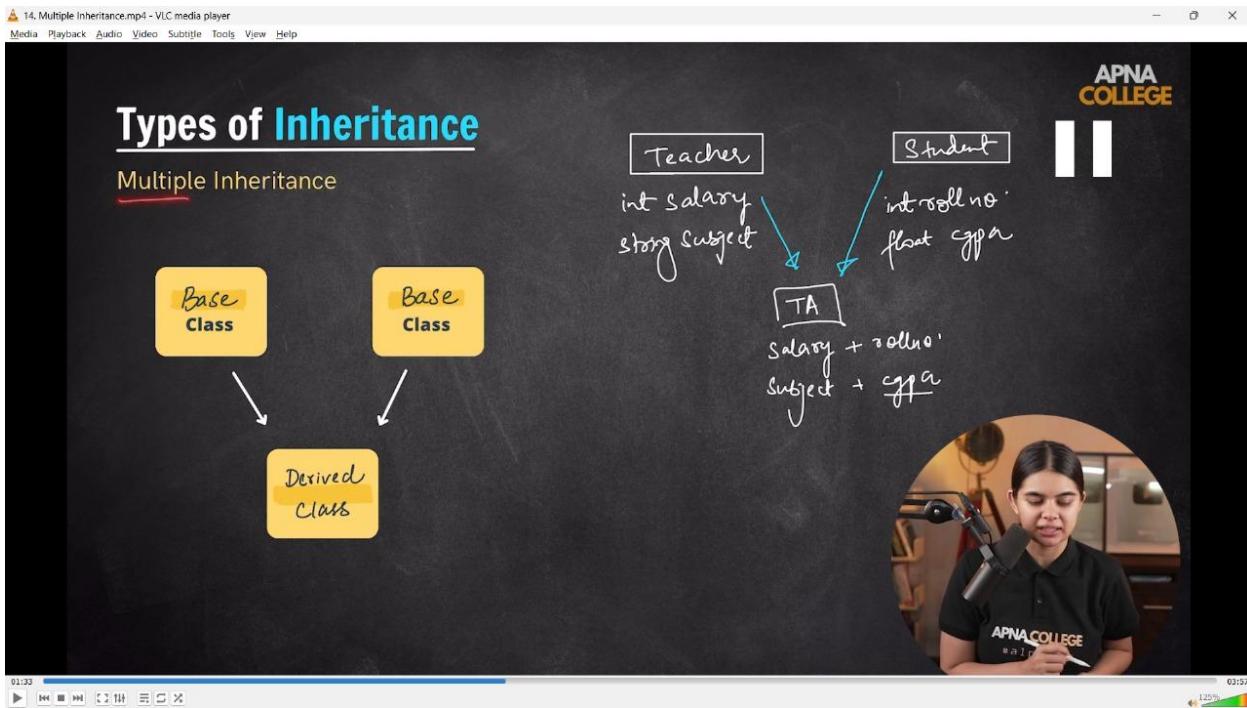
// Dog can also wipe out its tail everytim,e

// So, the colour of blood is - Greenish

// */
// }
// _____

```

#### //4.2.3) Multiple Inheritance -



```

// class Teacher

// {
// public:
//     int salary;

```

```
// string subject;
//};

// class Student
//{
// public:
// long rollno;
// float cgpa;
//};

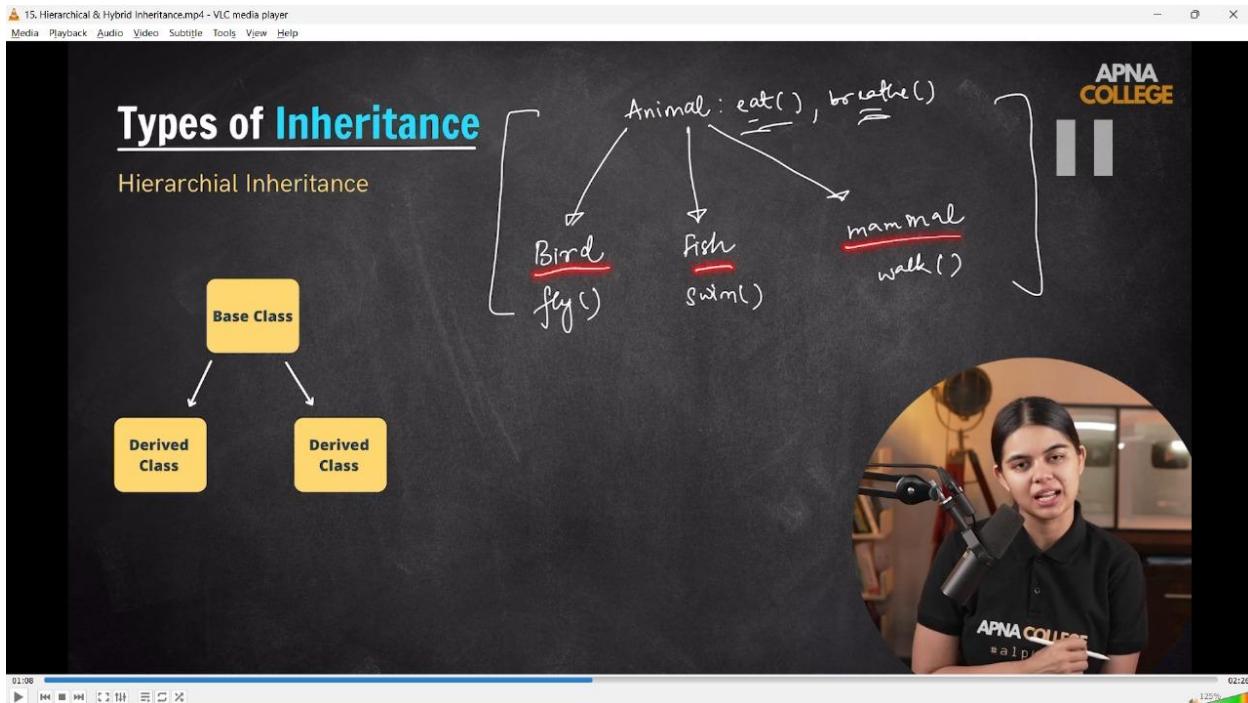
// class TA : public Teacher, public Student
//{
// public:
// string name;
//};

// int main()
//{
// TA ta1;
// ta1.name = "Shubham Mahajan";
// ta1.subject = "C++";
// ta1.rollno = 4088;
// ta1.cgpa = 7.5;

// cout << ta1.name << endl;//Shubham Mahajan
// cout << ta1.cgpa << endl;//7.5
// cout << ta1.subject << endl;//C++
// cout << ta1.rollno << endl;//4088
```

```
// }  
// _____
```

#### //4.2.4) Hierarchical Inheritance -



```
// class Animal  
// {  
// public:  
//     void Eat()  
//     {  
//         cout << "Can eat easily" << endl;  
//     }  
//     void Breathe()  
//     {  
//         cout << "Breathing as a regular process.." << endl;  
//     }  
//};
```

```
// class Bird : public Animal  
// {  
// public:  
//     void fly()  
//     {  
//         cout << "Can fly too.." << endl;  
//     }  
//};
```

```
// class Fish : public Animal  
// {  
// public:  
//     void swim()  
//     {  
//         cout << "Can swim in water..." << endl;  
//     }  
//};
```

```
// class Mammal : public Animal  
// {  
// public:  
//     void walk()  
//     {  
//         cout << "Can also walk regularly.." << endl;  
//     }  
//};
```

```
// };

// int main()
// {
//   Mammal m1;
//   m1.Breathe(); //Breathing as a regular process..
//   m1.Eat(); //Can eat easily
//   m1.walk(); //Can also walk regularly..

//   cout << endl;

//   Bird b1;
//   b1.Eat(); //Can eat easily
//   b1.Breathe(); //Breathing as a regular process..
//   b1.fly(); //Can fly too..

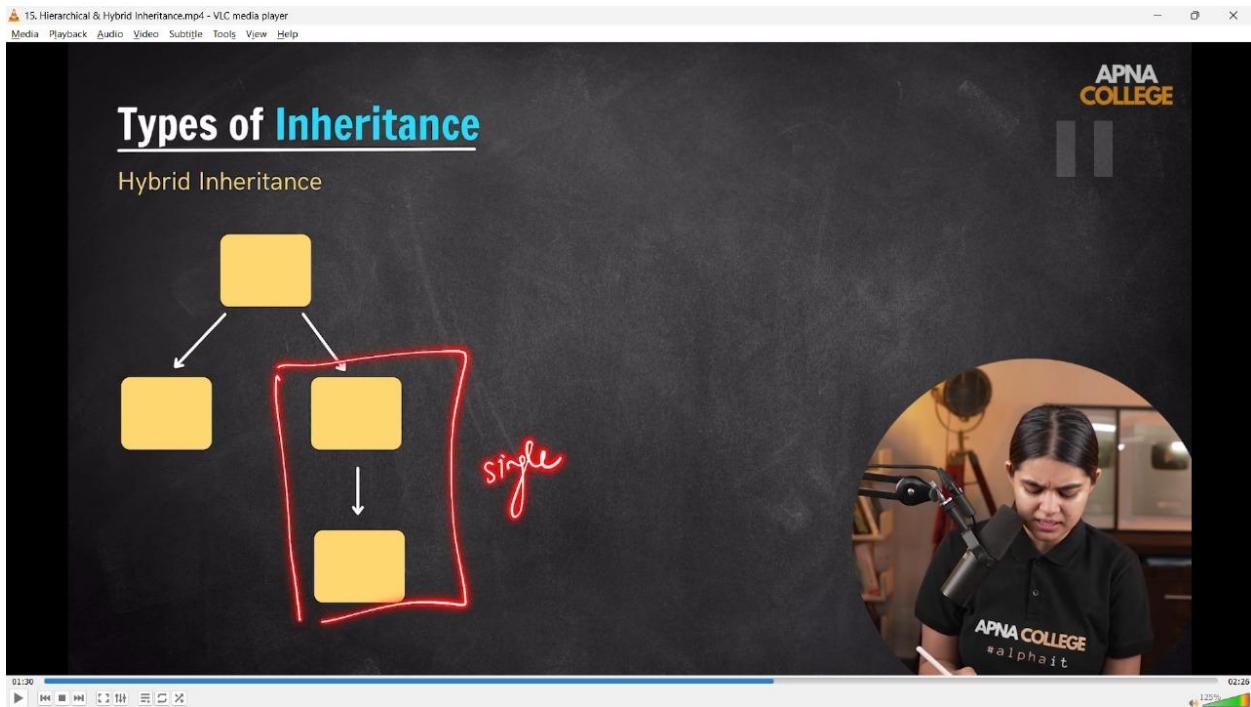
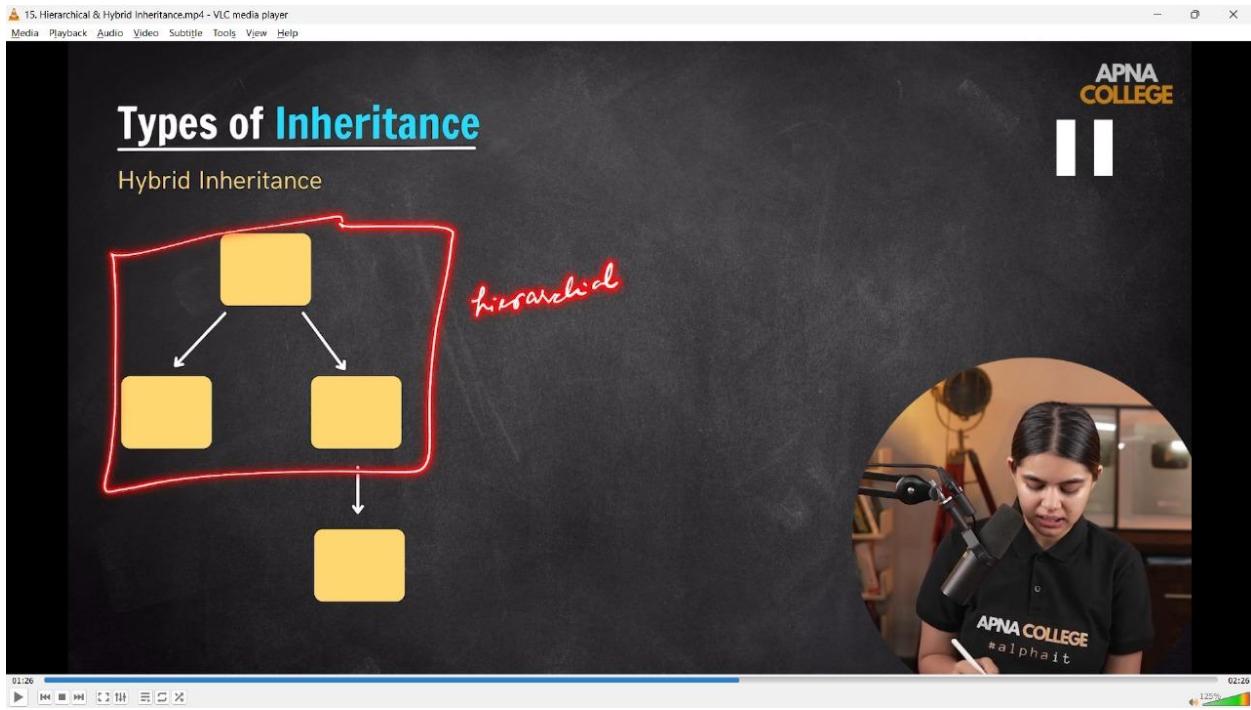
//   cout << endl;

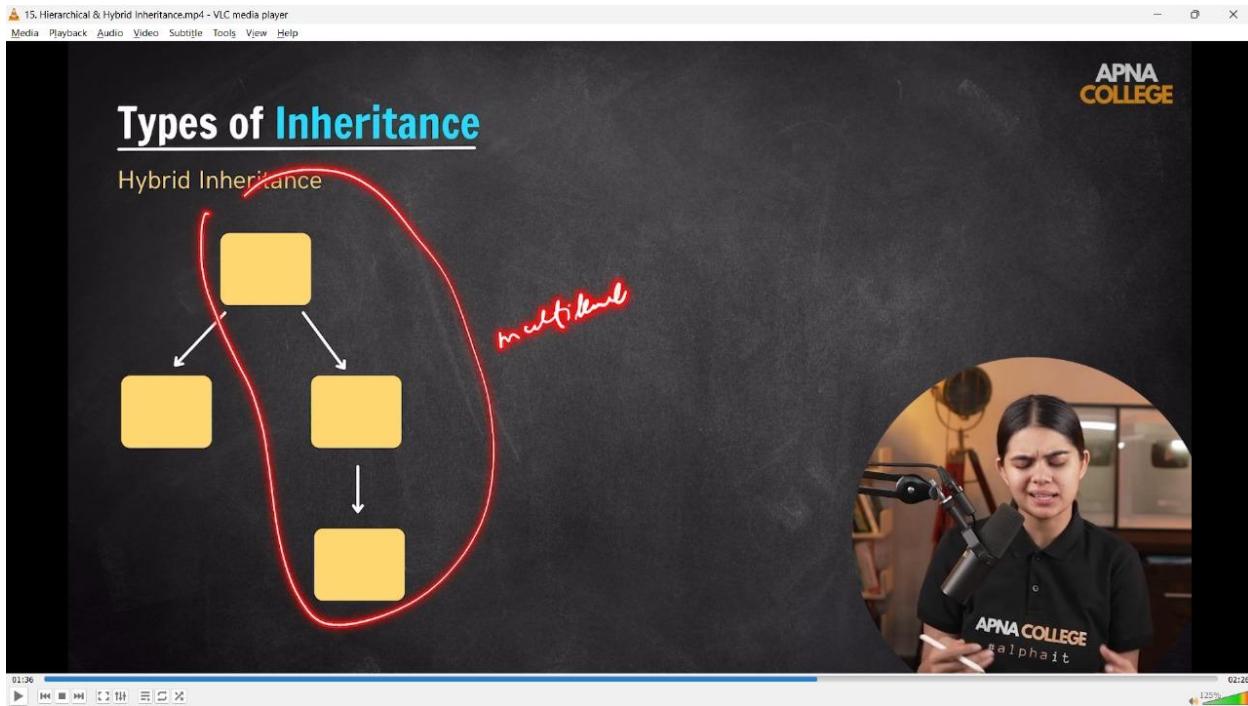
//   Fish f1;
//   f1.Breathe(); //Breathing as a regular process..
//   f1.swim(); //Can swim in water...

//   return 0;
/// // So, the clearly visible example of Hierarchical Inheritance whether multiple derived
// classed from the one base class
// }
```

//

### //4.2.5) Hybrid Inheritance -





```
// class Animal
//{
// public:
//     void Eat()
//     {
//         cout << "Can eat easily" << endl;
//     }
//     void Breathe()
//     {
//         cout << "Breathing as a regular process.." << endl;
//     }
//};

// class Fish : public Animal
//{

```

```
// public:  
// void swim()  
// {  
//     cout << "Can swim in water..." << endl;  
// }  
// };  
  
// class Turtle : public Fish  
// {  
// public:  
// void walkAndSwim()  
// {  
//     cout << "Can also walk as well as swim too.." << endl;  
// }  
// };  
  
// class Bird : public Animal  
// {  
// public:  
// void fly()  
// {  
//     cout << "Can fly too.." << endl;  
// }  
// };  
  
// int main()
```

```
// {  
//   Turtle t1;  
//   t1.Breathe(); // Breathing as a regular process..  
//   t1.Eat(); // Can eat easily  
//   t1.swim(); // Can swim in water...  
//   t1.walkAndSwim(); // Can also walk as well as swim too..  
  
// /*  - so clearly its an example if Hybrid Inheritance(as the combination of more than one inheritance) whether -  
// Single Inheritance - Class Fish is derived from Class Animal -  
// Multi Level Inheritance - Class Turtle is derived from class Fish which is further derived from Animal class  
// Hierarchical Inheritance - Class Bird is also derived from the class Animal  
// */  
  
// }  
// _____
```

## //5) Polymorphism -

### //5.1) Compiletime Polymorphism

The screenshot shows a VLC media player window with a slide titled "Polymorphism". The slide has a chalkboard background. At the top right, there is handwritten text: "constructor overloading" and the APNA COLLEGE logo. Below the title, there is handwritten text: "Forms" with a red arrow pointing down, followed by the definition: "Polymorphism is the ability of objects to take on **different forms** or behave in different ways **depending on the context** in which they are used." To the right of the text is a circular video frame showing a person speaking into a microphone. On the left side of the slide, there is a bulleted list: • Compile Time Polymorphism and • Run Time Polymorphism. To the right of the list, there is a diagram showing three arrows pointing from the text "Car()" to three separate "Car" constructor definitions: "Car()", "Car(name, color)", and "Car(name)".

#### 5.1.1) Function Overloading

The screenshot shows a VLC media player window with a slide titled "Compile Time Polymorphism". The slide has a chalkboard background. At the top right, there is the APNA COLLEGE logo. To the right of the title, there is handwritten text: "Point obj1", "obj1.show(25)", and "obj1.show("hello")". Below the title, there is a code snippet: 

```
class Point {  
    show(int)  
    show(string)  
}
```

 Arrows point from the handwritten text to the corresponding function signatures in the code. To the right of the code, there is a circular video frame showing a person speaking into a microphone. The video player interface at the bottom shows a progress bar from 01:18 to 02:24 and a 125% zoom level.

```

//5.1.1) Function Overloading - where same name fun with diff. arguements with different
values

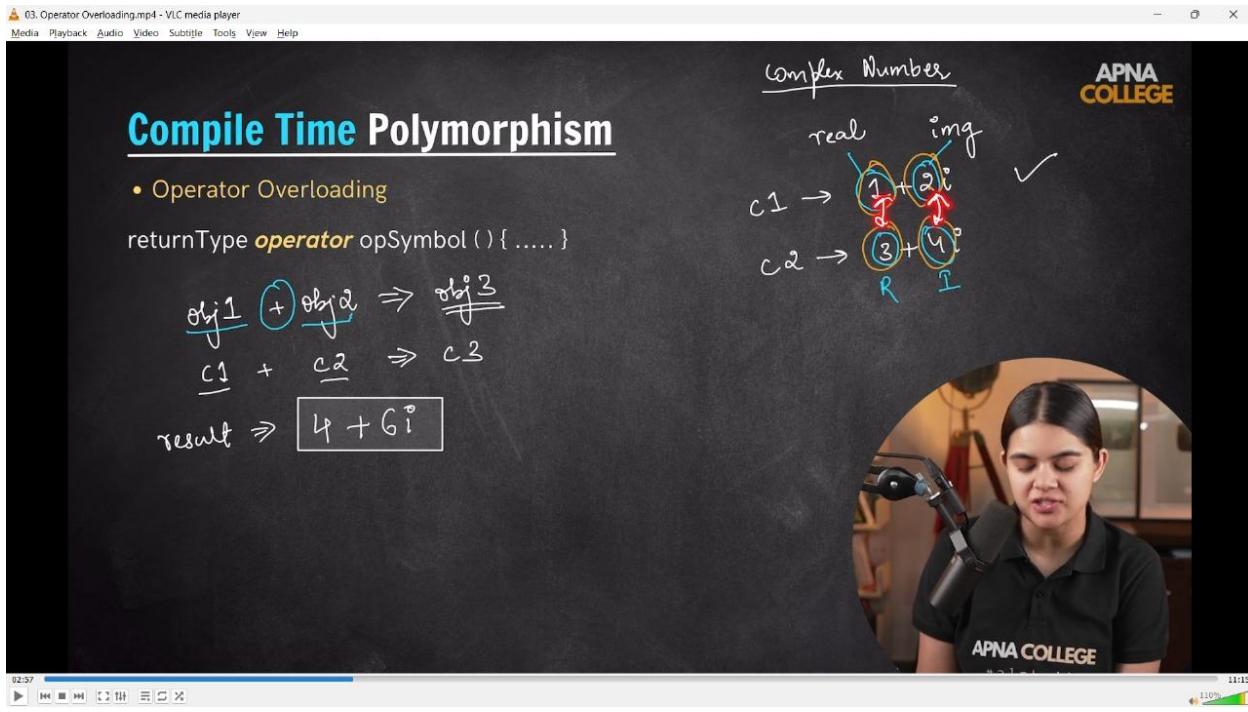
// class Print

//{
// public:
// void show(int x)
// {
//     cout<<"int : "<<x<<endl;
// }
// void show(string str)
// {
//     cout<<"string : "<<str<<endl;
// }
//}

// int main()
//{
// Print obj1;
// obj1.show(51);//int : 51
// obj1.show("ApnaCollege");//string : ApnaCollege
// /* So when we called the fun then int waale fun me int value pass hui & string waale fun
me string waali val
// So, in such a way in Function Overloading Compile Time Polymorphism is called]
// */
//}

```

**//5.1.2) Function Overloading -**



/\*

Oeprator Overloading - so as we've seen that - in constructor with the same name & diff. parameters - its called COnstrcutor Overloading

When same named function with different arguments are known as Function Overloadig

So similarly when sme operator but works differetn is called Operator Overloading

\*/

```
// class Complex
```

```
// {
```

```
//   int real;
```

```
//   int img;
```

```
// public:
```

```
//   Complex(int r, int i)
```

```
//   {
```

```
//     real = r;
```

```
//     img = i;  
// }  
  
// void ShowNum()  
// {  
//     cout << real << "+" << img << "i" << endl;  
// }  
  
// void operator + (Complex &c2)// Operator Overloading  
// {  
//     int resReal = this-> real + c2.real;  
//     int resImg = this->img + c2.img;  
//     Complex c3(resReal, resImg);  
//     c3.ShowNum();  
// }  
  
// void operator - (Complex &c2)// Operator Overloading  
// {  
//     int resReal = this-> real - c2.real;  
//     int resImg = this->img - c2.img;  
//     Complex c3(resReal, resImg);  
//     c3.ShowNum();  
// }  
//};
```

```
// int main()  
// {  
//     Complex c1(1, 2);  
//     Complex c2(3, 4);
```

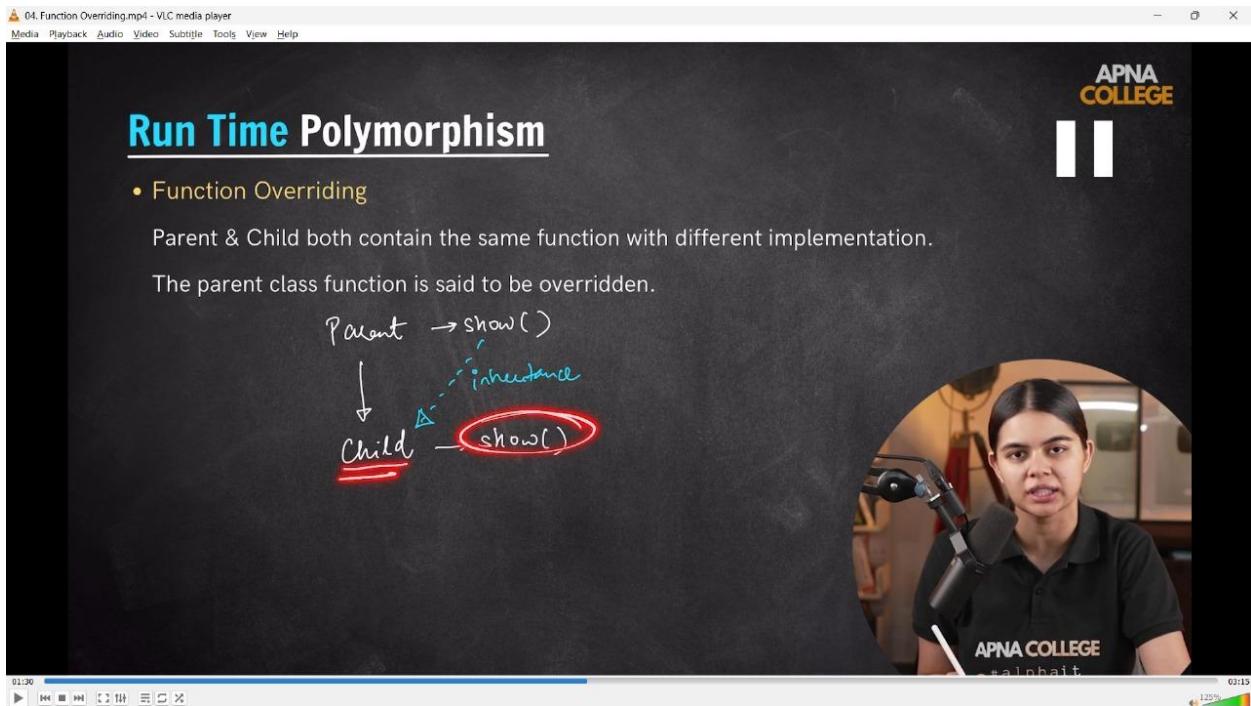
```

// c1.ShowNum(); //1+2i
// c2.ShowNum(); //3+4i
// c1+c2; //4+6i
// c1-c2; //-2+-2i
// }
// _____

```

## //5.2)Polymorphism -

### //5.2.1)Function Overriding -



//5.2.1)Function Overriding - when parent & child both contain the same fun wit different implementation. Parent & child dono classses me same fun h but diff datatypes ke liye diff implementations he , this is Function Overriding.

```

// class Parent
//{
// public:
// void show()

```

```

// {
//     cout << "Parent Class Show Function.." << endl;
// }
//};

// class Child : public Parent

// {

// public:

//     void show()

//     {

//         cout << "Chil CLass Show Function .." << endl;

//     }

// };



// int main()

// {

//     Child child1;//Created Object of child class

//     child1.show();//Chil CLass Show Function ..

// /* Overloading me ek hi class ke under dono fun ka name same hota h jabki

// OverRiding me ek calss inherited hoti h or usme fun ka name same hota he

// */

// }

// -----



//5.2.2)Virtual Functions -

```

05. Virtual Functions.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Run Time Polymorphism

- Virtual Functions
  - Virtual functions are Dynamic in nature.
  - Defined by the keyword "virtual" inside a base class and are always declared with a base class and overridden in a child class.
  - A virtual function is called during Runtime

APNA COLLEGE

#alphait

04:42 04:50 125%

05. Virtual Functions.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Run Time Polymorphism

- Virtual Functions

A virtual function is a member function that you expect to be redefined in derived classes.

Speed: 1.00

APNA COLLEGE

#alphait

00:51 04:50 125%

```
// class Parent  
// {  
// public:  
//     virtual void hello()
```

```

// {
//     cout << "parent Hello " << endl;
// }
//};

// class Child : public Parent

// {

// public:

//     void hello()

//     {

//         cout << "Child classs hello --" << endl;

//     }

// };

// int main()

// {

//     Child child1;//child class ka oonject bnaya he

//     Parent *ptr; // Parent class k liye pointer define kiya he

//     ptr = &child1; // run time waali bindng - Parent class ke pointer ko child class me point kra rhe h

//     ptr->hello(); // Child classs hello -- parent class ke pointer se child class ke hello fun ko call kra]tya

//     // Child Func>>Inheritance

// }

// _____

```

## **6) Abstraction –**

06. Abstraction.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

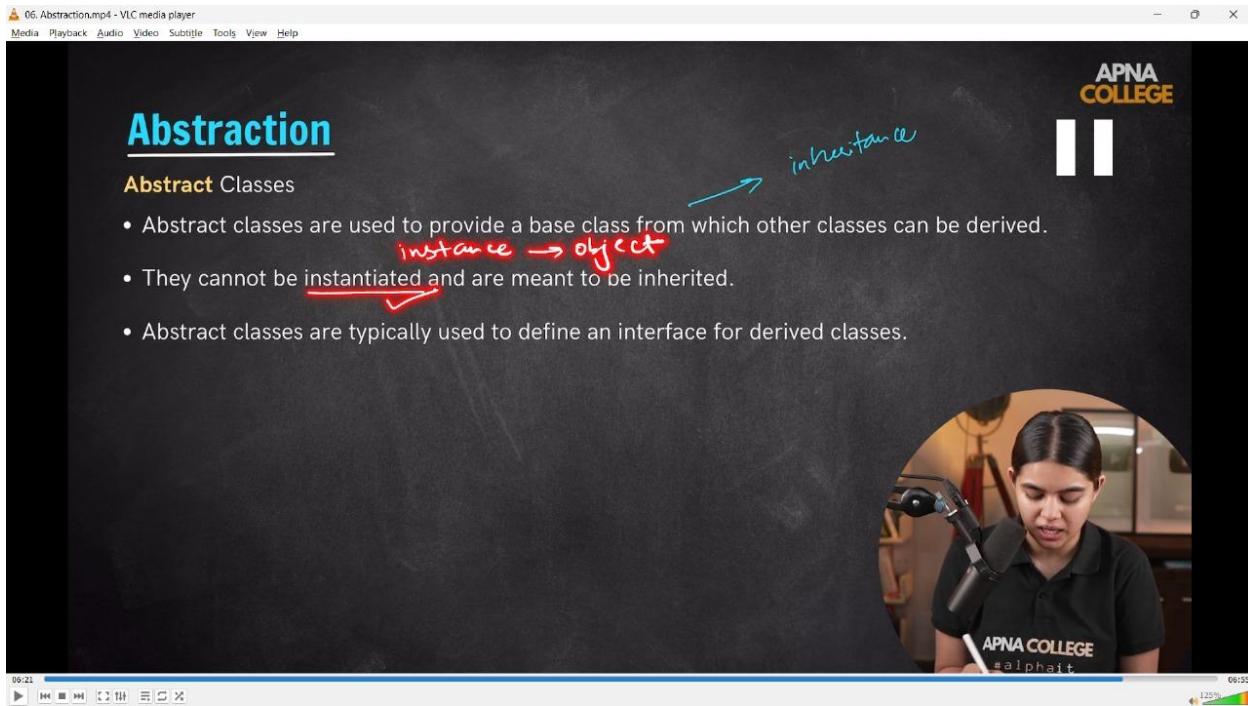
## Abstraction

**Abstract Classes**

- Abstract classes are used to provide a base class from which other classes can be derived.
- They cannot be instantiated and are meant to be inherited.
- Abstract classes are typically used to define an interface for derived classes.

instance → object

instance inheritance



06. Abstraction.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Abstraction

Hiding all unnecessary details & showing only the important parts

**Abstract Classes & Pure Virtual Functions**

access specifies

private  
protected  
public

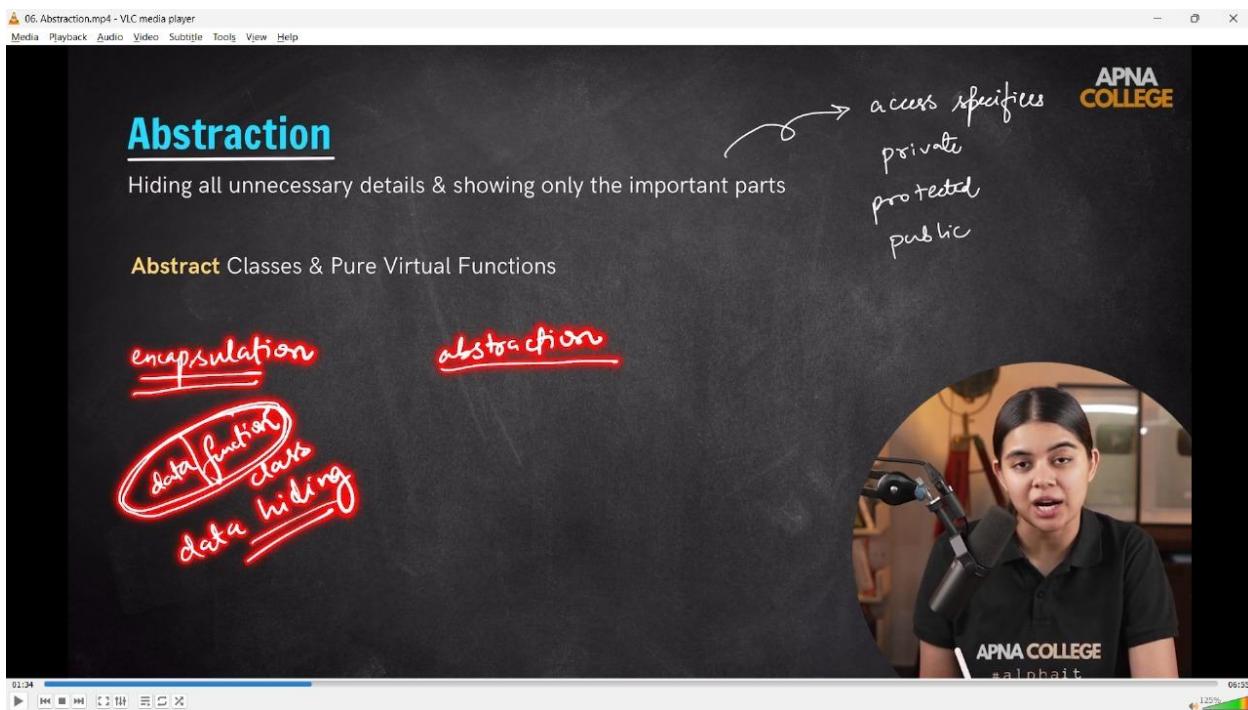
encapsulation

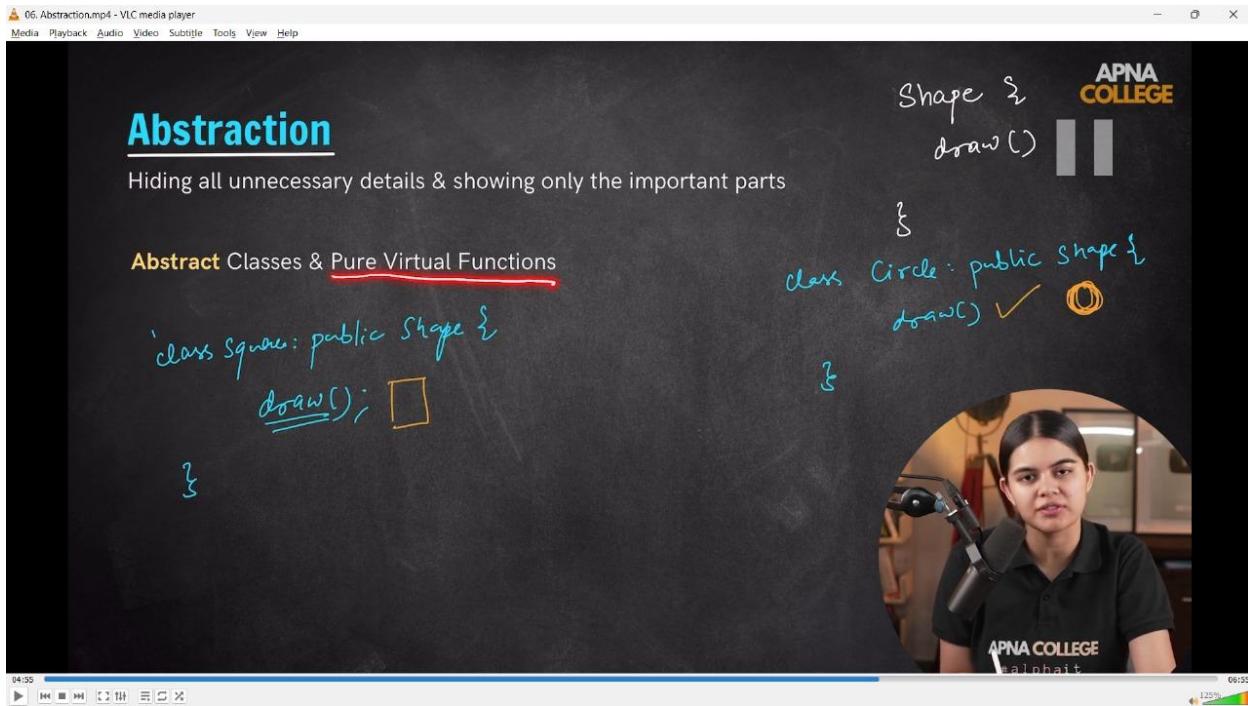
abstraction

data function

data class

data hiding





//6) Abstraction - Hiding all unnecessary details & showing only the important parts

```
/* Access Modifier is a way which defined to abstraction. Because for unnecessary detail it can
be hide through PRIVATE & PROTECTED while necessary details can be shown through PUBLIC
*/
```

//6.1) Pure Virtual Function -

```
// class Shape
//{
// public:
//     virtual void draw() = 0; // Pure virtual function - enforces derived classes to implement
//     draw()
//};
```

```
// class Circle : public Shape
```

```
// {
```

```
// public:  
// void draw() override  
// {  
//     cout << "Draw Circle -" << endl;  
// }  
// };  
  
// class Square : public Shape  
// {  
// public:  
// void draw() override  
// {  
//     cout << "Draw Square -" << endl;  
// }  
// };  
  
// int main()  
// {  
//     Circle cir1;  
//     cir1.draw();  
  
//     Square squ1;  
//     squ1.draw();  
  
//     return 0;  
// }
```

//

## 7) Static Keyword –

### // 1) For functions -

The screenshot shows a VLC media player window displaying a video slide. The title of the slide is "Static Keyword". Below the title, there is a bulleted list: "• Static Variables". To the right of the list, there is a hand-drawn diagram on a chalkboard. The diagram shows a vertical stack of boxes representing function scopes. The top box is labeled "main", and below it is a larger box labeled "counter". Inside the "counter" box, the variable "static count = 0" is written. To the right of the diagram, there is some handwritten code: "counter();", "counter();", "counter();", and "counter();". The APNA COLLEGE logo is visible in the top right corner of the slide. The VLC interface includes a toolbar at the bottom with playback controls like play, stop, and volume, and a status bar at the bottom right showing the duration as 09:49 and a 125% zoom level.

This screenshot shows the same VLC media player window as the previous one, but with several red annotations overlaid on the slide content. Red lines have been drawn through the handwritten code "counter();", "counter();", "counter();", and "counter();". Instead, the slide now displays the numbers "1 & 3" in red. The rest of the slide content, including the title "Static Keyword", the bullet point "• Static Variables", and the explanatory text about static variables being created once per function lifetime, remains unchanged. The APNA COLLEGE logo is still present. The VLC interface at the bottom is identical to the previous screenshot.

```
// void Counter()  
// {  
//     int count = 0;  
//     count++;  
//     cout << "Count is : " << count << endl;  
// };  
  
// int main()  
// {  
//     Counter(); // Count is : 1  
//     Counter(); // Count is : 1  
//     Counter(); // Count is : 1  
//     /* here w/o declaring static keyword in the function, the count variable created again and  
again 3 times in the memory, but when we use static fun then the variable only creates once  
and updates value  
// */  
// }  
  
// void Counter()  
// {  
//     static int count = 0;  
//     count++;  
//     cout << "Count is : " << count << endl;  
// };  
  
// int main()  
// {
```

```
// Counter();//Count is : 1  
// Counter();//Count is : 2  
// Counter();//Count is : 3  
// }
```

---

## // 2) For class

The screenshot shows a VLC media player window with the title "08. Static Keyword.mp4 - VLC media player". The video content is a slide from a presentation. The slide has a dark background and features the title "Static Keyword" in large blue text. Below the title is a bulleted list: "• Static Variables". To the right of the list are three white circles containing handwritten text: "eg 1 int x=0", "eg 2 int x=0", and "eg 3 int x=0". In the bottom right corner of the slide, there is a logo for "APNA COLLEGE". A circular video frame in the bottom right corner shows a person speaking into a microphone. The VLC interface at the bottom includes standard controls like play/pause, volume, and a progress bar indicating the video is at 00:51 of 09:49.

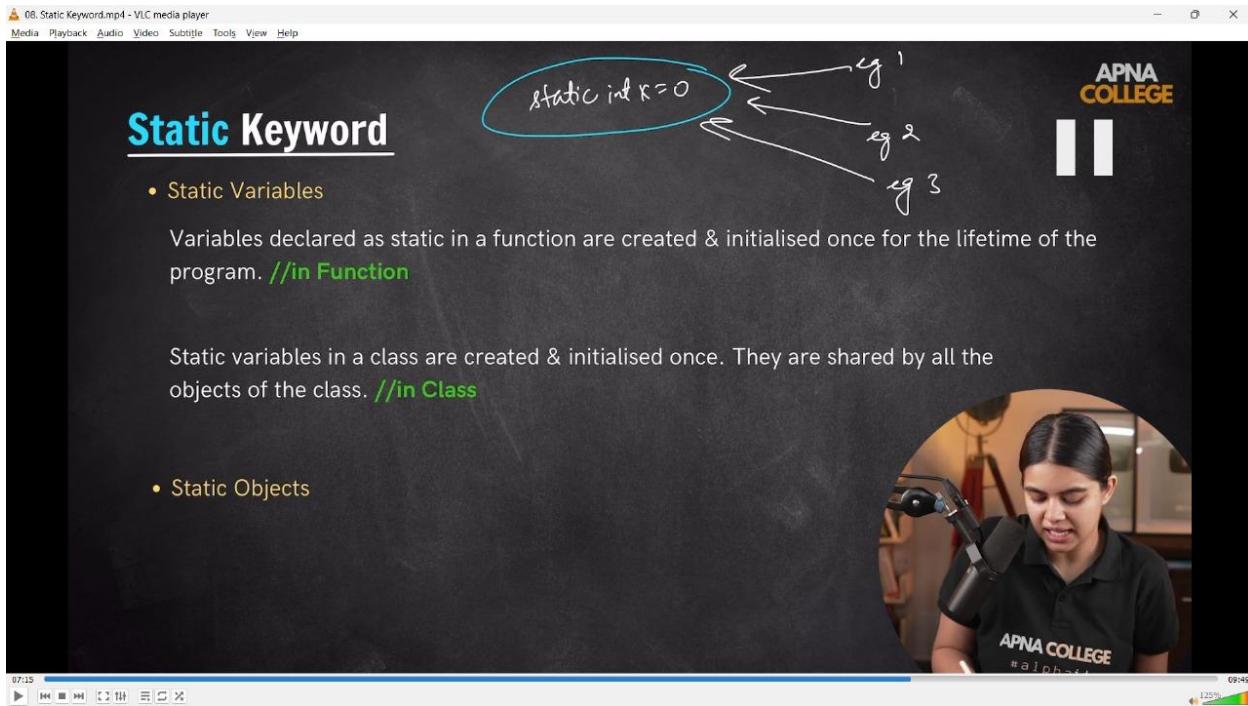
**Static Keyword**

- Static Variables

Variables declared as static in a function are created & initialised once for the lifetime of the program. //in Function

Static variables in a class are created & initialised once. They are shared by all the objects of the class. //in Class

- Static Objects



```
// class Example
```

```
// {  
// public:  
//     int x = 0;  
// };
```

```
// int main()  
// {  
//     Example eg1;  
//     Example eg2;  
//     Example eg3;
```

```
//     cout << eg1.x++ << endl; // 0  
//     cout << eg2.x++ << endl; // 0  
//     cout << eg3.x++ << endl; // 0  
// }
```

```
// when we use static keyword for the class then it is created for all the objects assignes witht  
the same class
```

```
// when we use static keyword for the class, then for it need to done the declaration(::) of  
variable witht he class
```

```
// class Example
```

```
// {
```

```
// public:
```

```
// static int x;
```

```
//};
```

```
// int Example::x = 0; // declaration of x in the example class
```

```
// int main()
```

```
// {
```

```
// Example eg1;
```

```
// Example eg2;
```

```
// Example eg3;
```

```
// cout << eg1.x++ << endl; // 0
```

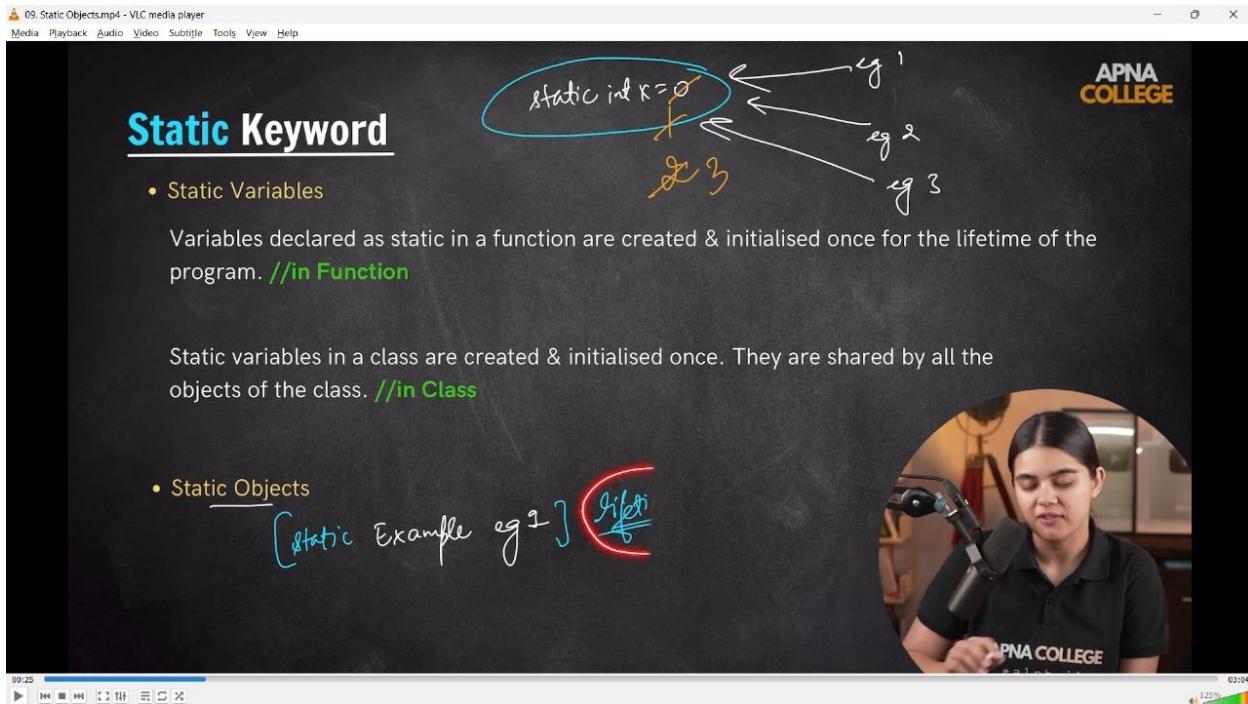
```
// cout << eg2.x++ << endl; // 1
```

```
// cout << eg3.x++ << endl; // 2
```

```
// }
```

```
// _____
```

```
// 3) For obejcts - Stattic keyword is for lifetime
```



```
// class Example
```

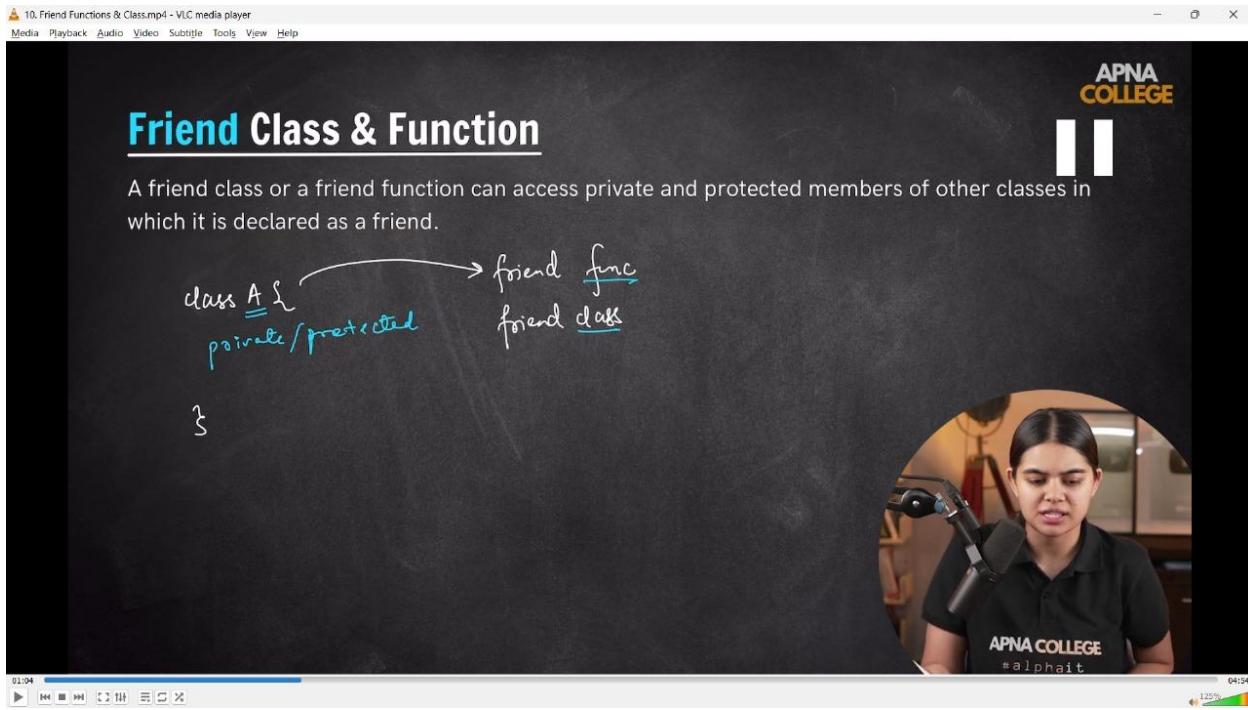
```
// {  
// public:  
//     Example()  
//     {  
//         cout << "Constructor - " << endl;  
//     }  
// }
```

```
// ~Example()  
// {  
//     cout << "Destructor - " << endl;  
// }  
// };  
// int main()  
// {
```

```
// int a = 0;  
// if (a == 0)  
// {  
//     Example eg1;  
// }  
// cout << "Code Ending - " << endl;  
/**/  
// Constructor -  
// Destructor -  
// Code Ending -
```

```
// Bt what happens when we use static keyword-  
// */  
// if (a == 0)  
// {  
//     static Example eg1;  
// }  
// cout << "Code Ending - " << endl;  
/**/  
// Constructor -  
// Destructor -  
// Code Ending -  
// */  
// }
```

## 8) Friend Class & Friend Function –



//7) Friend Class & Friend Function - A friend who has over ovt info can be reveal outside

```
// class A  
  
// {  
//     string secret = "Personal Data";  
// };  
  
// class B  
  
// {  
//     public:  
//         void showSecret(A &obj)  
//         {  
//             // cout<<obj.secret<<endl;  
//         }  
//         // error: 'std::__cxx11::string A::secret' is private within this context
```

```
// // this is invalid and showing error, not accesible to priovate class's data
// // So, for avoiding this we can do ki B class ko class A ka friend bna diya
// //};

// int main()
//{
// A a1;
// B b1;

// b1.showSecret(a1);
//}

// -----
// class A
//{
// string secret = "Personal Data";
// friend class B;// cslls b has been become friend for class A
//};

// class B
//{
// public:
// void showSecret(A &obj)
// {
//     cout << obj.secret << endl;//Personal Data
// }
}
```

```

// };

// int main()
// {
//     A a1;
//     B b1;

//     b1.showSecret(a1);
// }
// _____

```

### // Practice Q's -

11. Practice Qs.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Practice Qs**

Predict the Output

```

class A {
public:
    A() { std::cout << "Constructor A" << std::endl; }
    ~A() { std::cout << "Destructor A" << std::endl; }
};

class B : public A {
public:
    B() { std::cout << "Constructor B" << std::endl; }
    ~B() { std::cout << "Destructor B" << std::endl; }
};

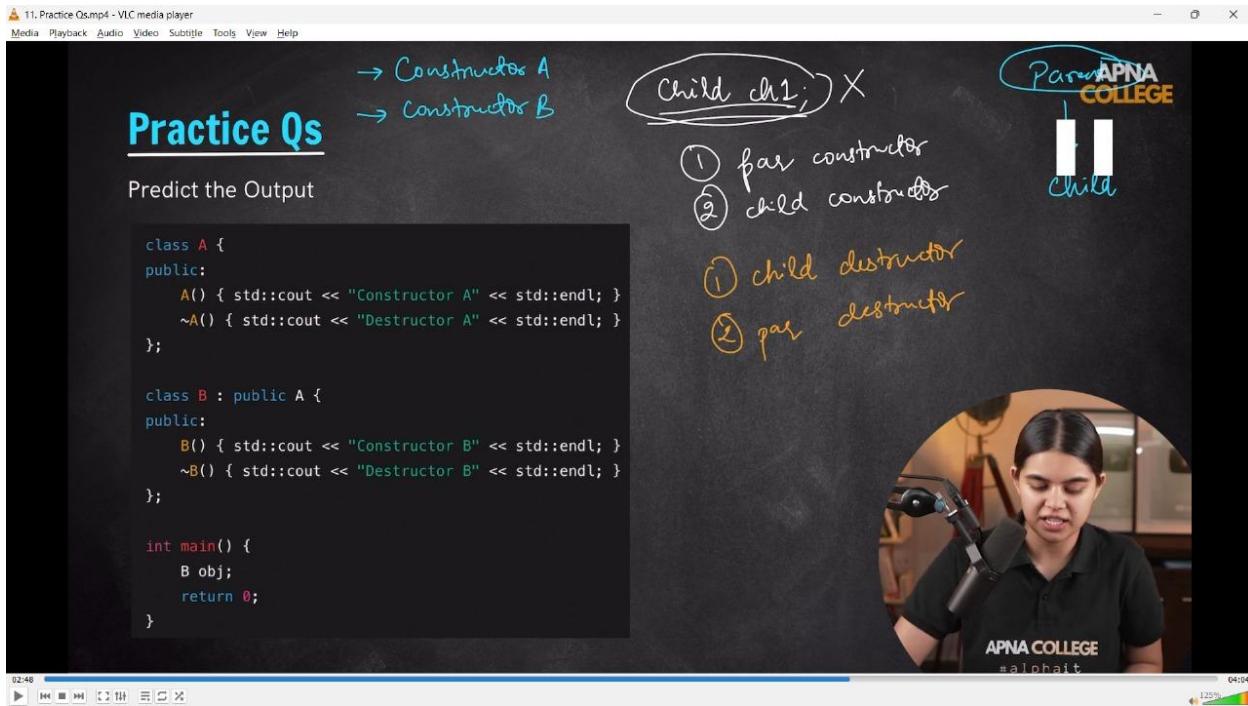
int main() {
    B obj;
    return 0;
}

```

Child ch1;

① far constructor  
② child co.

Parin APNA COLLEGE



/\*

Predict the Output

```
class A {  
public:  
    A() { std::cout << "Constructor A" << std::endl; }  
    ~A() { std :: cout << "Destructor A" << std :: endl; }  
};
```

```
class B : public A {  
public:  
    B() { std::cout << "Constructor B" << std::endl; }  
    ~B() { std :: cout << "Destructor B" << std :: endl; }  
};
```

```
};

int main() {
B obj;
return 0;
}

}

/*
// class Parent
//{
// public:
// Parent()
// {
//     cout << "Constructor Parent Class " << endl;
// }

// ~Parent()
// {
//     cout << "Dectorator Parent Class " << endl;
// }
//};

// class Child : public Parent
//{
// public:
// Child()
// {

```

```

//      cout << "Constructor Child CLass " << endl;
//  }
// ~Child()
// {
//      cout << "Dectorctor Child CLass " << endl;
// }
//};

// int main()
//{
//  Child ch1; // Object of child class, first call always go to Parent Class Contructor then it calls
// to Child Class COnstructor

// /* O/p -
// Constructor Parent Class
// Constructor Child CLass
// */

// /* O/p - // In case if Descructor, first call always go to Child Class Destructor then it calls to
// Parent Class Constructor

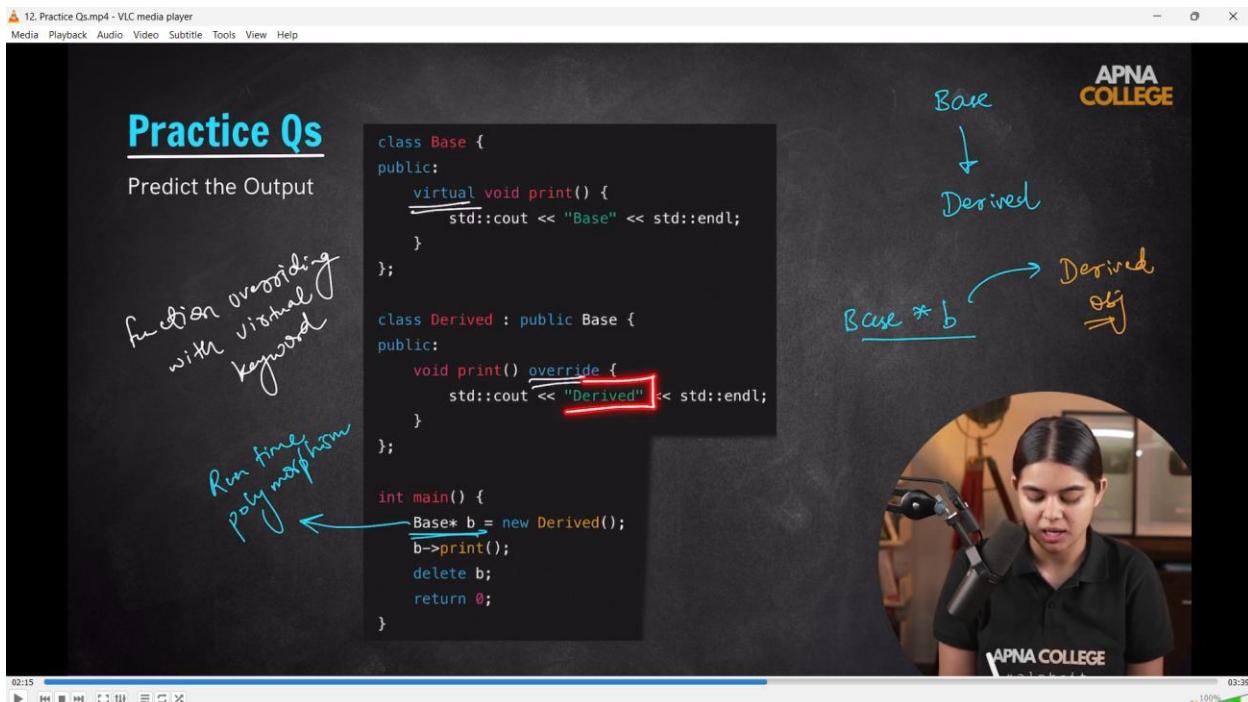
// Dectorctor Child CLass
// Dectorctor Parent Class
// */

// /*
//  - So order for constructor & discructor calling for parent & Child Classes
// i) Constructor of Parent Class
// ii) Constructor of Child Class
// iii) Dewstructor of Parent Class

```

```
// iv) Destructor of Child Class  
// */  
// }
```

**//Qun - 2 :-**



// \_\_\_\_\_

\_\_\_\_\_

## **[9] Recursion –**

1) Recursion Basic Understanding-

2) Recursion Qn's -

2.1) Print the numbers in descending Order

2.2) Stack Overflow -

2.3) Sum of N-Natural Numbers -

2.4) Print Nth Fibonacci Number -

2.5) Check if Array is Sorted or Not -

2.6) First Occurrence -

2.6.1) Last Occurrence -

2.7) X to the power N -

3) Tiling problem

4) Remove Duplicates from the String -

5) Friends Pairing Problem -

6) Binary String Problem –

## 9) Recursion -

1) Recursion - When a function repeatedly calls itself is called Recursion

02. Dive Deep into Recursion.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

APNA COLLEGE

## Recursion

It is a method of solving computational problems where the solution depends on solutions to smaller instances of the same problem.

- ① Base Case / smallest problem ✓
- ② kaam? in each func call
- ③ inner func

19:09 13:20

125%

A woman is speaking into a microphone in a circular inset at the bottom right.

01. What is Recursion.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

APNA COLLEGE

## Recursion

when a function repeatedly calls itself

Diagram illustrating recursion:

- Recursive call: A main function box points to a func function box.
- Call stack: A sequence of func function boxes connected by arrows labeled "def" (definition) pointing to the left and "ret" (return) pointing to the right.
- Base case: An arrow points from the end of the call stack to a small box labeled "end point".
- Diagram on the right shows a recursive call structure with nested boxes and arrows.

04:23 06:43

125%

A woman is speaking into a microphone in a circular inset at the bottom right.

01. What is Recursion-.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Recursion

when a function repeatedly calls itself

loops

1  
2  
3  
4

APNA COLLEGE

06:39 06:43 125%

02. Dive Deep into Recursion.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Recursion

understanding using math

$$f(x) = x^2 \quad \text{function}$$

$$f(f(x)) = (x^2)^2 \Rightarrow f(f(x)) = (2^2)^2 = 4^2 = 16$$

int func( int x )  
return x\*x;

APNA COLLEGE

01:30 13:20 125%

02. Dive Deep into Recursion.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Recursion**

understanding using math

$$0! = 1$$

$$1! = 1$$

$$\frac{5!}{4!} = \frac{5 \times 4 \times 3 \times 2 \times 1}{4 \times 3 \times 2 \times 1} = \frac{5 \times 4!}{4 \times 3!}$$

$$2! = \frac{3 \times 2 \times 1}{3 \times 2 \times 1} = 3 \times 2!$$

$$n! = n \times (n-1) \times (n-2) \dots \times 1$$

$$f(n) = n!$$

$$f(n) = n \times f(n-1) \quad \text{] Recurrence Relation}$$

$$(n-1) \times f(n-2)$$

$$(n-2) \times f(n-3)$$

$$\vdots$$

APNA COLLEGE

04:46 13:20 125%

02. Dive Deep into Recursion.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Recursion**

understanding using math

$$f(n) = n \times f(n-1) \quad \text{] Recurrence Relation}$$

$$\frac{n=5}{f(5)} = \frac{5 \times f(4)}{4} \quad 5! = 5 \times 4!$$

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

$$4 \times f(3) \quad 6$$

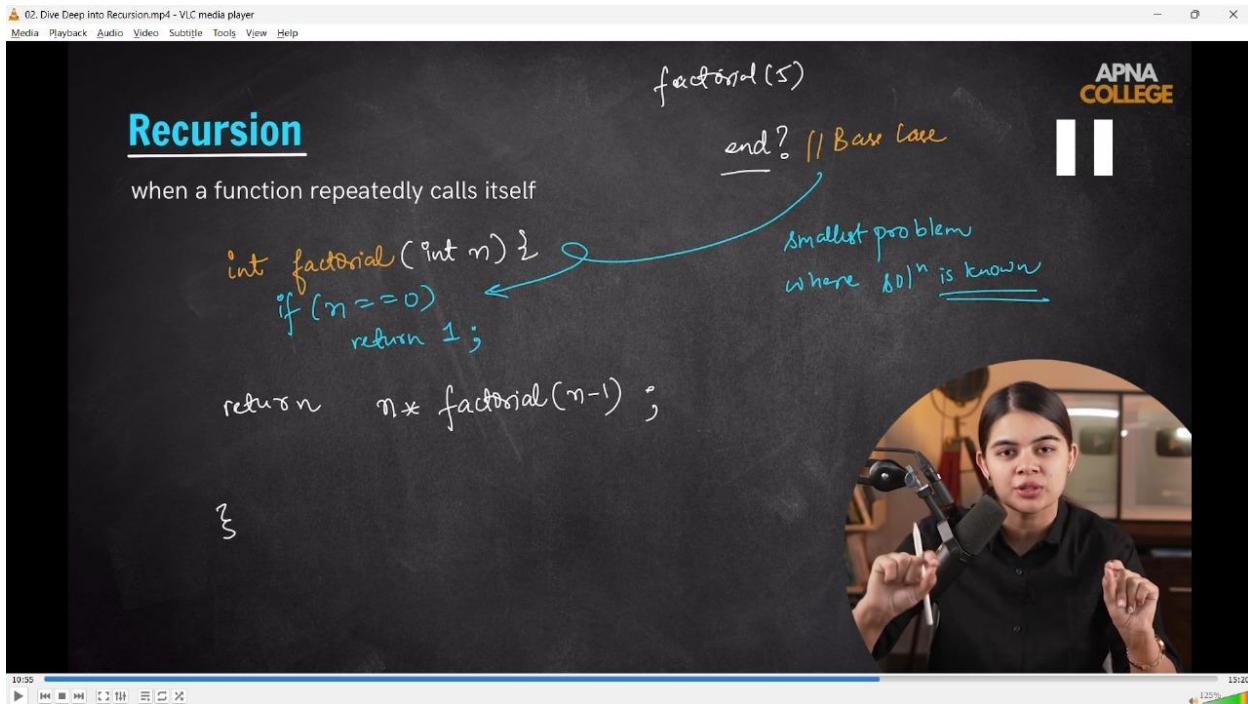
$$3 \times f(2) \quad 2$$

$$2 \times f(1) \quad 1$$

$$1 \times f(0) \quad 1$$

smallest pos b) APNA COLLEGE

09:46 13:20 125%



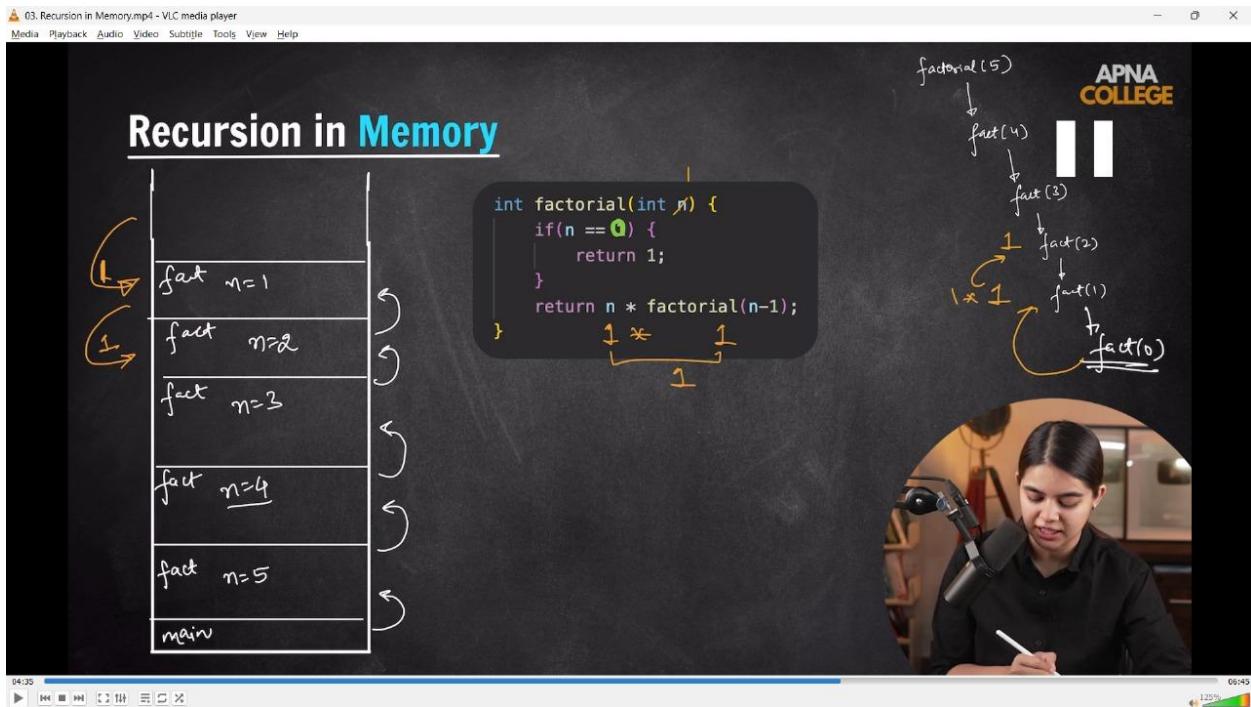
```
// int factorial(int n)  
// {  
//     if (n == 0 || n == 1)//Base Case  
//     {  
//         return 1;  
//     }  
//     return n * factorial(n - 1);//Working condition with smallest condition  
// }  
  
// int main()  
// {  
//     // Recursive fun for facvtorial -  
//     int n;  
//     cout << "What is the n's value - " << endl;  
//     cin >> n;
```

```
// cout << factorial(n) << endl;//120  
// cout << factorial(6) << endl;//720  
// cout << factorial(7) << endl;//5040  
// cout << factorial(8) << endl;//40320
```

```
// }
```

```
// _____
```

## Recursion In Memory –



03. Recursion in Memory.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Recursion in Memory

```

int factorial(int n) {
    if(n == 0) {
        return 1;
    }
    return n * factorial(n-1);
}

```

$5 \times 24 = 120$

APNA COLLEGE

fact(5) → fact(4) → fact(3) → fact(2) → fact(1) → fact(0) ← fact(5)

main → 120

## 2) Recursion QUns -

### //2.1) Print the numbers in descending Order

04. Numbers in Decreasing Order.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Print numbers in decreasing order

From n to 1

$\text{print}(1)$

$\text{print}(n)$

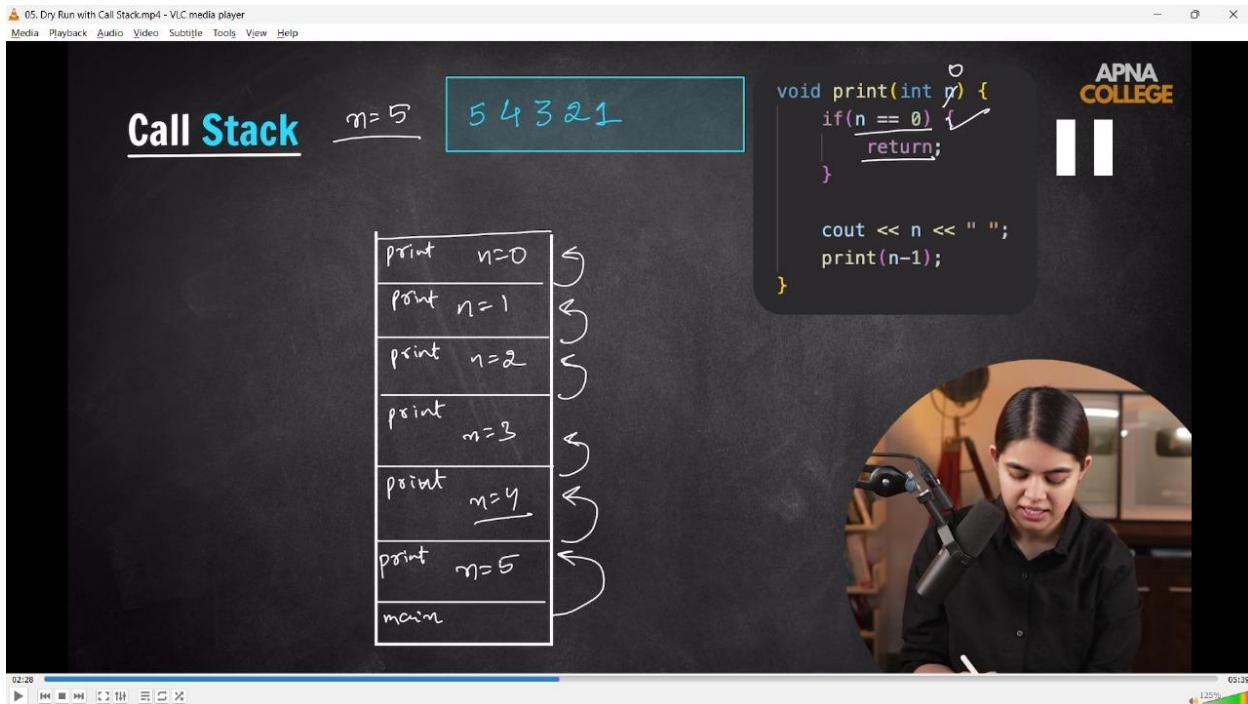
assume →  $\text{print}(n-1)$

kaam  $\text{cout} < n$

Base Case  $n=0$  return

5 to 1 → 4 to 1 → 3 to 1 → 2 to 1 → 1 to 1 → 0 BC

print(5) → print(4) → print(3) → print(2) → print(1) → print(0)



```
// void func()

// {
//   cout << "Function called ....working" << endl;
//   func();
// }

/// // Gives Segmentation fault at a time when memory is fully filled

// int main()

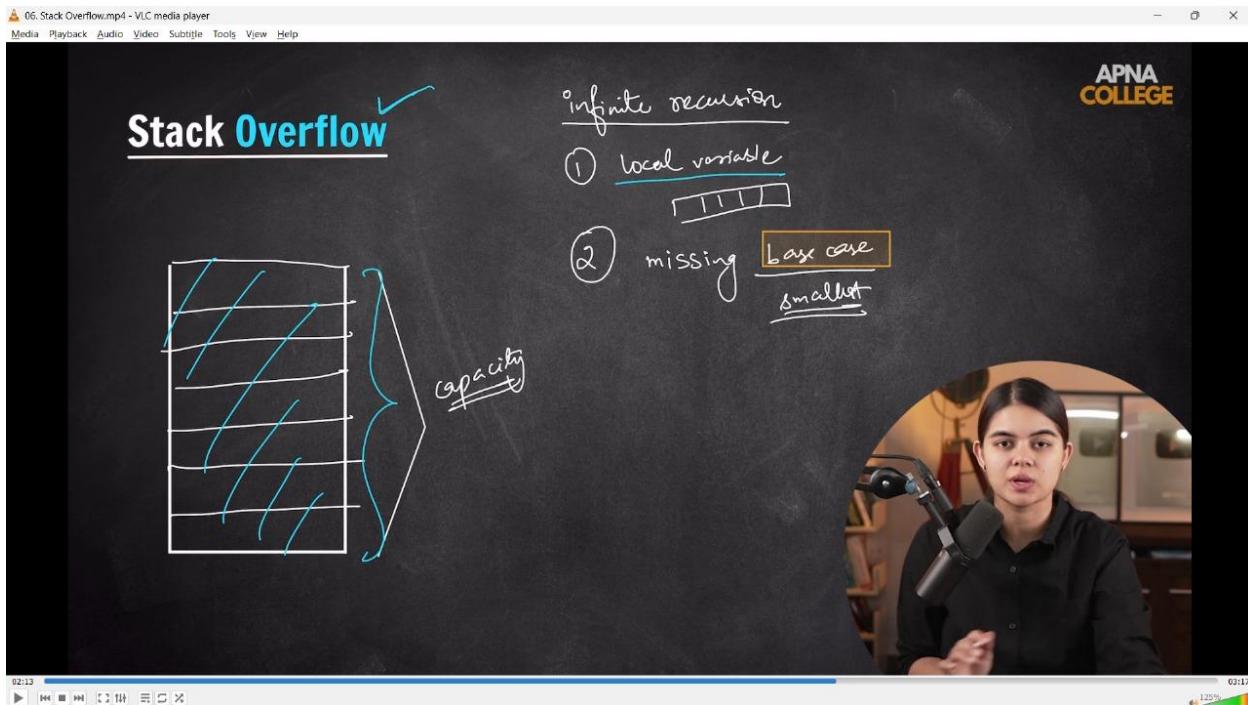
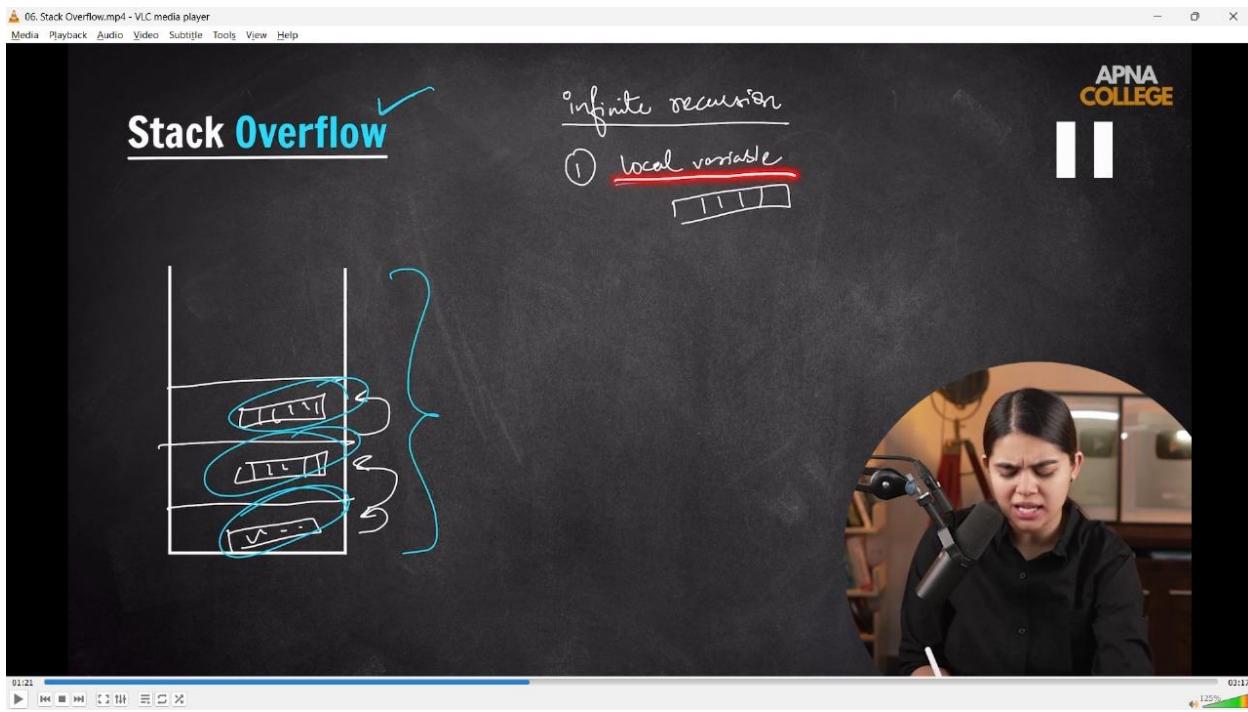
// {
//   return 0;
// }

// /*

// The function func() is recursively calling itself without a base condition, which leads to infinite
recursion, ultimately overflowing the call stack, resulting in a segmentation fault.

// */
// 
```

## //2.2) Stack Overflow –



```
// void func()  
// {  
//   cout << "Function called ....working" << endl;
```

```

// func();
// }

/// Gives Segmentation fault at a time when memory is fully filled

// int main()

//{
// return 0;
//}

// */

// The function func() is recursively calling itself without a base condition, which leads to infinite
recursion, ultimately overflowing the call stack, resulting in a segmentation fault.

// */
// _____

```

### //2.3)Sum of N-Natural Numbers -

07. Sum of N Natural Numbers.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Sum of N Natural Numbers**

$n = 5$

$\text{sum}(n) = n + \text{sum}(n-1)$

recurrence relation

$\text{sum}(5) = 5 + \text{sum}(4) \quad 10$

$= 5 + 4 + \text{sum}(3) \quad 6$

$= 5 + 4 + 3 + \text{sum}(2) \quad 3$

$= 5 + 4 + 3 + 2 + \text{sum}(1)$

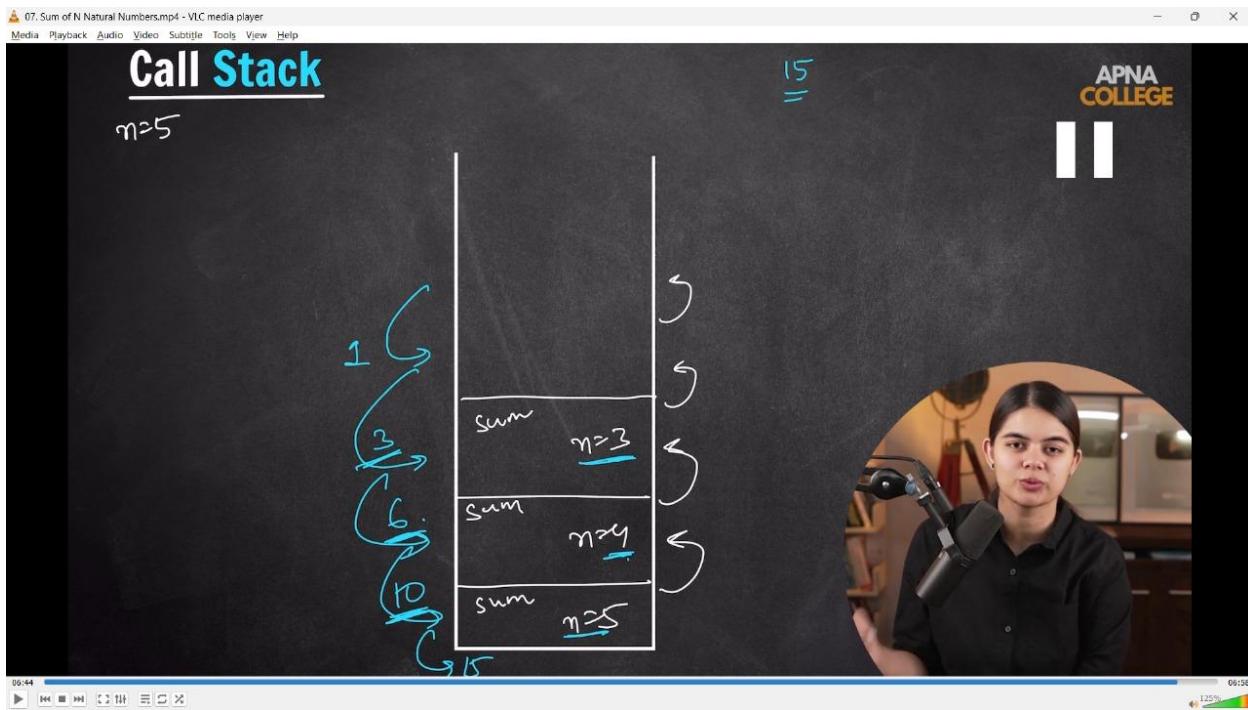
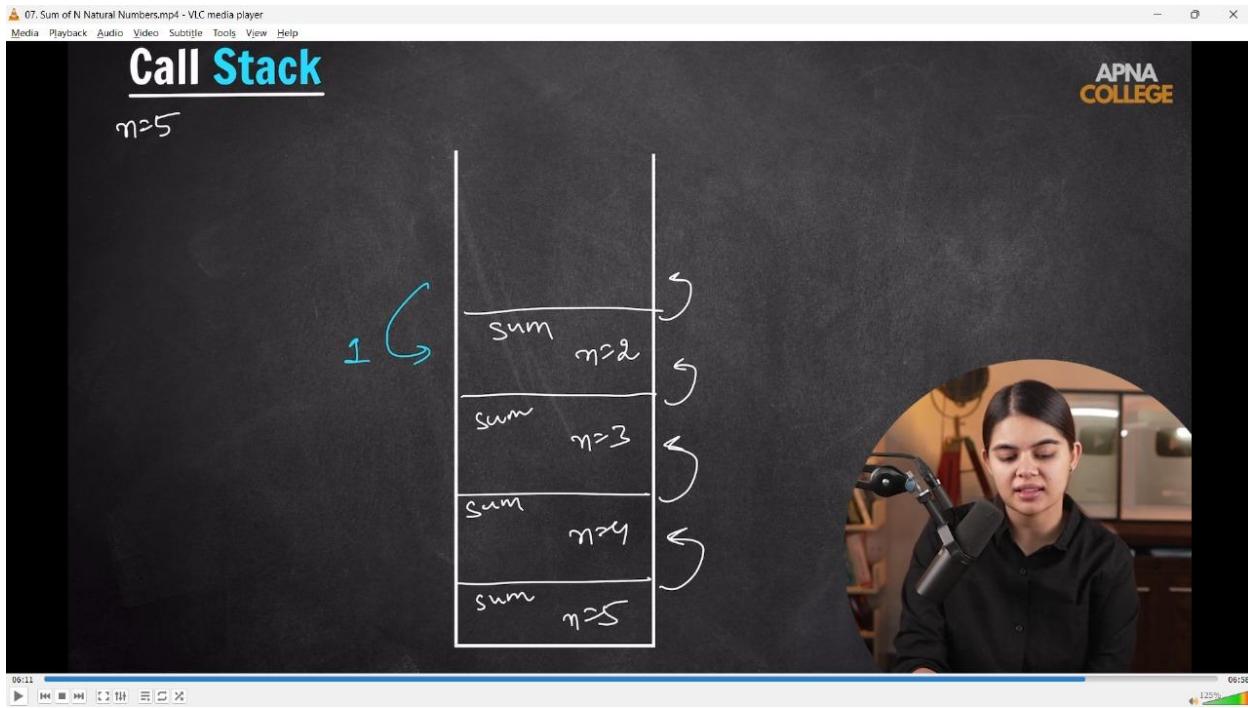
$= 15$

① **foor** **APNA COLLEGE**  
n +  $\text{fun}(n-1)$

② **kaam**

③ **base case** smallest problem  
 $n=1$   
return 1

03:36 06:59 125%



```
// int Sum(int n)
//{
//    if (n == 1)
//        {
```

```

//      return 1;

// }

//  return n + Sum(n - 1);

// }

// int main()

// {

//   cout<<Sum(5)<<endl;//15

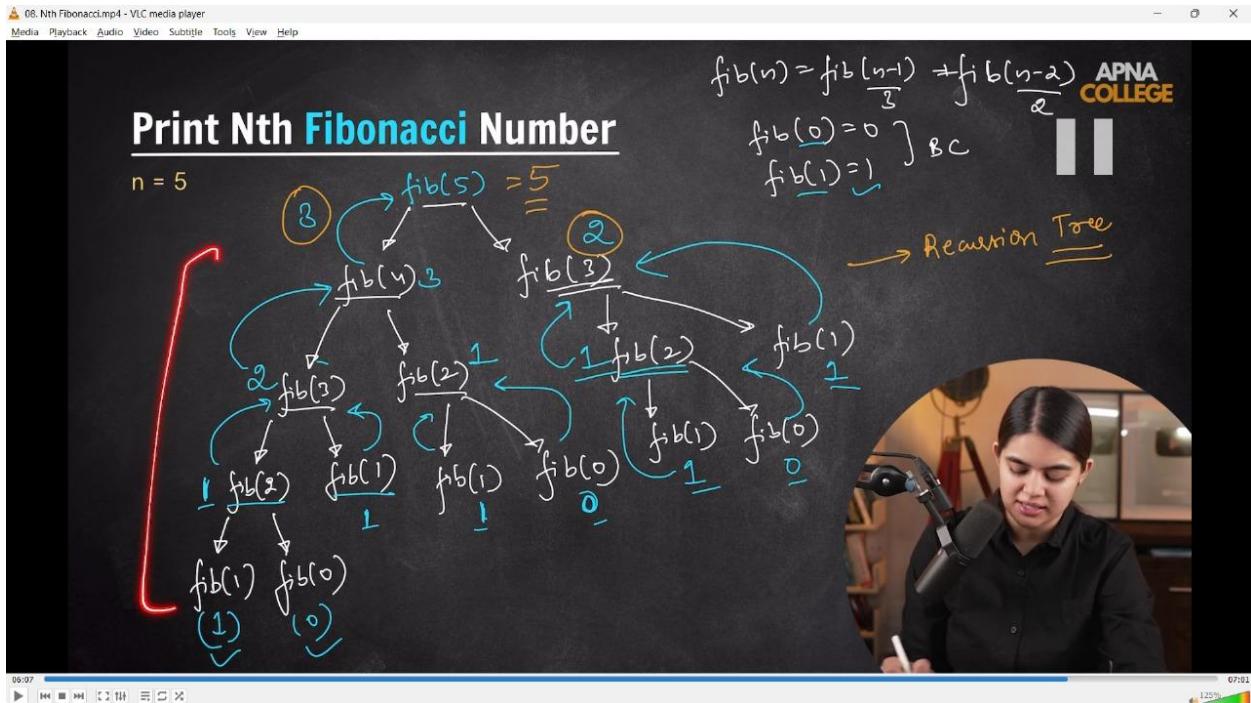
//   cout<<Sum(6)<<endl;//21

//   cout<<Sum(10)<<endl;//55

// }

// _____
```

#### //2.4) Print Nth Fibonacci Number -



06. Nth Fibonacci.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Print Nth Fibonacci Number

$n = 5$

$[5]$

$$\boxed{\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)}$$
$$\text{fib}(5)$$

①  $\text{fib}(n-1) + \text{fib}(n-2)$

②

06:49 07:01  
125%

06. Nth Fibonacci.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Print Nth Fibonacci Number

$n = 5$

$[5]$

$$\boxed{\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)}$$
$$\text{fib}(5)$$

①  $\text{fib}(n-1) + \text{fib}(n-2)$

②

06:57 07:01  
125%

```
// int fibonacci(int n)  
// {  
//     if (n == 0 || n == 1)  
//     {
```

```
//      return n;  
// }  
//  return fibonacci(n - 1) + fibonacci(n - 2);  
// }  
// int main()  
// {  
//   cout << fibonacci(0) << endl; // 0  
//   cout << fibonacci(1) << endl; // 1  
//   cout << fibonacci(2) << endl; // 1  
//   cout << fibonacci(3) << endl; // 2  
//   cout << fibonacci(4) << endl; // 3  
//   cout << fibonacci(5) << endl; // 5  
//   cout << fibonacci(6) << endl; // 8  
//   cout << fibonacci(7) << endl; // 13  
//   cout << fibonacci(8) << endl; // 21  
//   cout << fibonacci(9) << endl; // 34  
//   cout << fibonacci(10) << endl; // 55  
// }  
// _____
```

**//2.5) Check if Array is Sorted or Not -**

10. Check if Array Sorted.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Check if Array Sorted**

$\text{arr} = [1, 2, 3, 4, 5]$  ascending

$i$   $\underline{\text{arr}, n}$   $\rightarrow \underline{\text{arr}, n-1}$   $\rightarrow \underline{\text{arr}, 1}$

$\text{arr}[i] \leq \text{arr}[i+1]$

① smaller problem  $\underline{\text{fun}(n-1)}$

APNA COLLEGE

02:06 08:21 125%

10. Check if Array Sorted.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Check if Array Sorted**

$i=0$   $\underline{\text{arr}, n}$  ascending

$\text{arr} = [1, 2, 3, 4, 5]$

①  $\text{arr}[i] > \text{arr}[i+1]$  return false ] kaam

②  $\text{isSorted}(\text{arr}, n, i) = \text{kaam}$   $\text{isSorted}(\text{arr}, n, i+1)$

③ BC  $\Rightarrow$  if  $i == n-1$  return true;

bool  $\text{isSorted}(\text{arr}, n, i)$

- if ( $i == n-1$ ) return true;
- [ if ( $\text{arr}[i] > \text{arr}[i+1]$ ) return false;
- [ return  $\text{isSorted}(\text{arr}, n, i+1)$ ;

APNA COLLEGE

05:36 08:21 125%

```
// bool isSortedArray(int arr[], int n, int i)
//{
//    if (i == n - 1)
//    {

```

```
//      return true;  
// }  
// if (arr[i] > arr[i + 1])  
// {  
//     return false;  
// }  
// return isSortedArray(arr, n, i + 1);  
// }  
  
// int main()  
// {  
//     int arr1[5] = {1, 2, 3, 4, 5}; // Sorted: 1  
//     int arr2[5] = {1, 2, 4, 3, 5}; // Unsorted : 0  
  
//     cout<<isSortedArray(arr1,5,0)<<endl;//1  
//     cout<<isSortedArray(arr2,5,0)<<endl;//0  
// }  
// _____  
//2.6) First Occurrence -
```

11. First Occurrence.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## First Occurrence

WAF to find first occurrence of an element in a vector.

$\text{arr} = [1, 2, 3, 3, 3, 4] \quad \text{target} = 3$

$i=0 \quad i=1 \quad i=2 \quad i=3 \quad i=4 \quad i=5 \quad i=6$

$\boxed{1 \ 2 \ 3 \ 3 \ 3 \ 4}$

$\text{ans} = 2$

$\boxed{i = n}$

$\boxed{\text{return } -1}$

$\boxed{\text{FO}(\text{arr}, i)}$

$\boxed{\text{if}(n == i) \text{return } -1}$

$\boxed{\text{if}(\text{arr}[i] == \text{target}) \text{return } i;}$

$\boxed{\text{return } \underline{\underline{\text{FO}(\text{arr}, i+1)}}}$

APNA COLLEGE

04:49 08:00 125%

```
// int firstOccurance(vector<int> arr, int i, int target)// phle apne liye check kr rhe h pphr abad waale index ke liye
```

```
// {
//   if (i == arr.size())
//   {
//     return -1;
//   }
```

```
//   if (arr[i] == target)
//   {
//     return i;
//   }
//   return firstOccurance(arr, i + 1, target);
//}

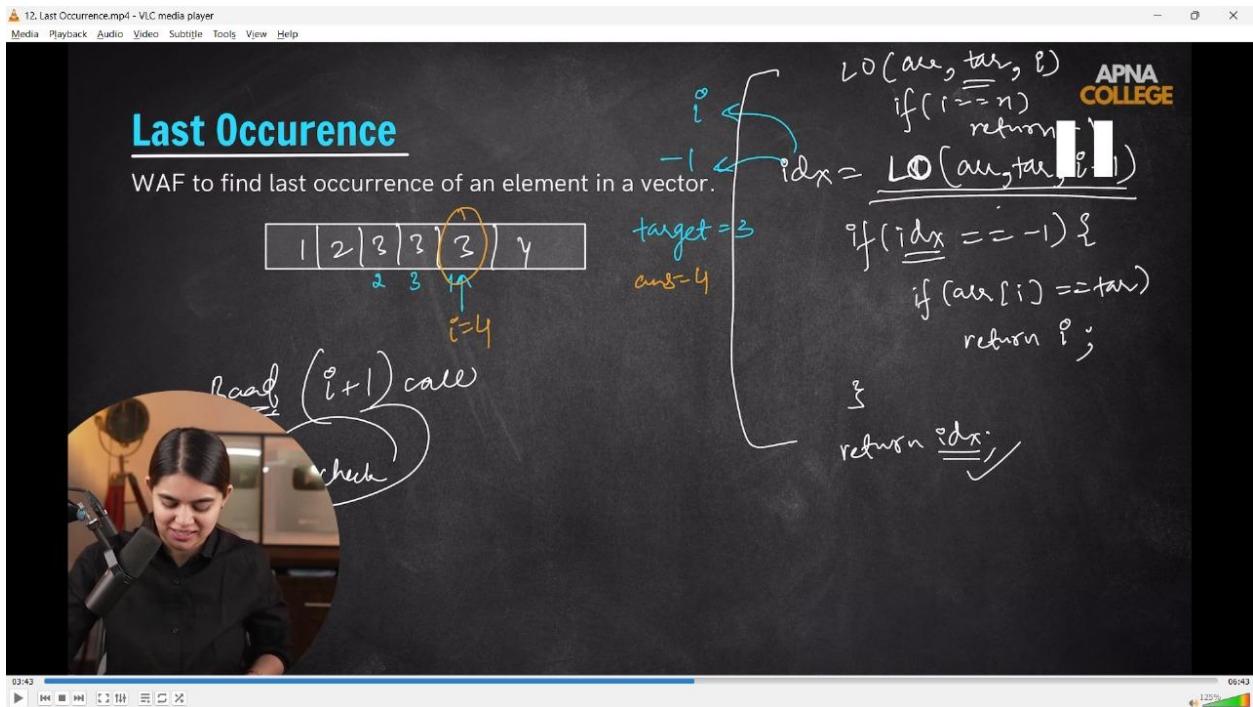
// int main()
```

```

// {
//   vector<int> arr = {1, 2, 3, 3, 3, 4, 5, 8, 8, 2, 9, 10};
//   cout << firstOccurrence(arr, 0, 3) << endl; //2
//   cout << firstOccurrence(arr, 2, 3) << endl; //2
//   cout << firstOccurrence(arr, 0, 8) << endl; //7
// }
// _____

```

### //2.6.1) Last Occurrence -



```

// int lastOccurrence(vector<int> arr, int target, int i)
// {
//   if (i == arr.size())
//   {
//     return -1;
//   }

```

```

// int idxFound = lastOccurance(arr, target, i + 1); //last occ. me phle baad wale k liye check kr
rhe h phr apne liye

// if (idxFound == -1)

// {

//     if (arr[i] == target)

//     {

//         return i;

//     }

// }

// return idxFound;

// }

// int main()

// {

//     vector<int> arr = {1, 2, 3, 3, 3, 4, 5, 8, 8, 2, 9, 10};

//     cout << lastOccurance(arr, 3, 0) << endl; // 4 from last occ.

//     cout << lastOccurance(arr, 2, 0) << endl; // 9

//     cout << lastOccurance(arr, 8, 7) << endl; // 8

//     cout << lastOccurance(arr, 45, 0) << endl; // -1

// }

// _____

```

**//2.7) X to the power N -**

Print X to the power N

$x^n \Rightarrow x=2, n=10 \Rightarrow 2^{10} = 1024$

$x^n \rightarrow$  n odd:  $2^5 \Rightarrow 2 * 2^2 * 2^2 \Rightarrow 2 * [2^{\frac{n_1}{2}} * 2^{\frac{n_2}{2}}]$

$\rightarrow$  n even:  $2^4 \Rightarrow 2^2 * 2^2 \Rightarrow 2^{\frac{n_1}{2}} * 2^{\frac{n_2}{2}}$

$2^{\frac{n_1}{2}} * 2^{\frac{n_2}{2}} \rightarrow 2^{\frac{n_1+n_2}{2}} = 2^{\frac{n}{2}} \Rightarrow 2^n$

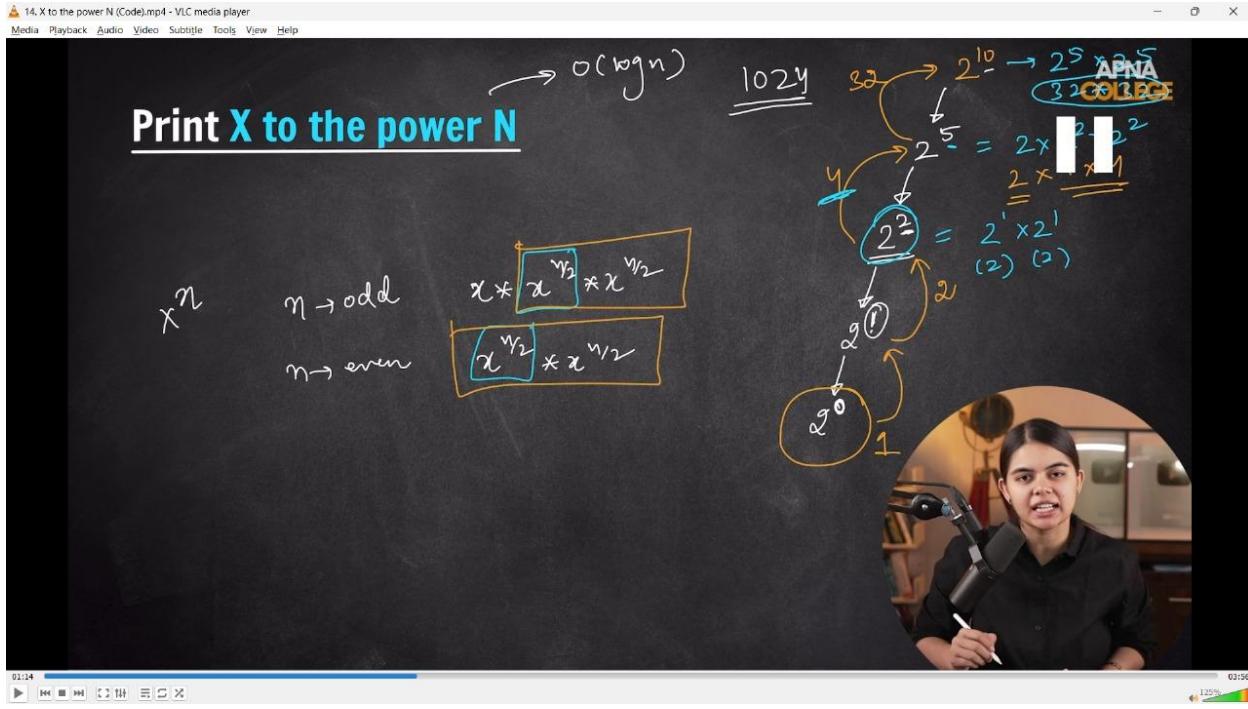
APNA COLLEGE

Print X to the power N

$n=10$

$1024 \rightarrow 2^{10} \rightarrow 2^5 * 2^5 \rightarrow 2^5 = 2 * 2^2 * 2^2 \rightarrow 2^2 = 2 * 2^1 \rightarrow 2^1 = 2$

$\log(n)$  (Binary Search)



```
// int pow(int x, int n)

// {
//   if (n == 0)
//   {
//     return 1;
//   }

//   int halfPow = pow(x, n / 2);
//   int halfPowSquare = halfPow * halfPow;

//   if (n % 2 != 0)
//   {
//     return x * halfPowSquare;
//   }

//   return halfPowSquare;
```

```

// }

// int main()
// {
//   cout << pow(2, 5) << endl;//32
//   cout << pow(2, 10) << endl;//1024
//   cout << pow(3, 4) << endl;//81
//   cout << pow(5, 5) << endl;//3125

```

// // T.C - Well optimized approach with TC O(log n)

```

// }

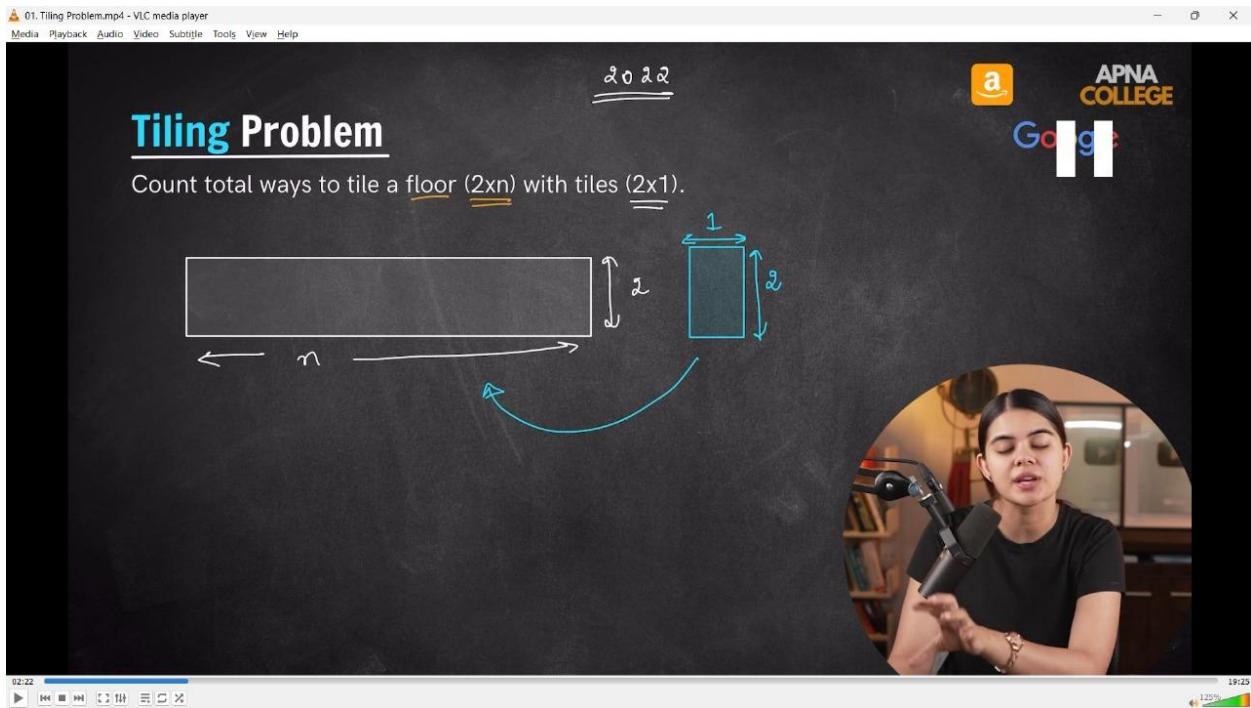
```

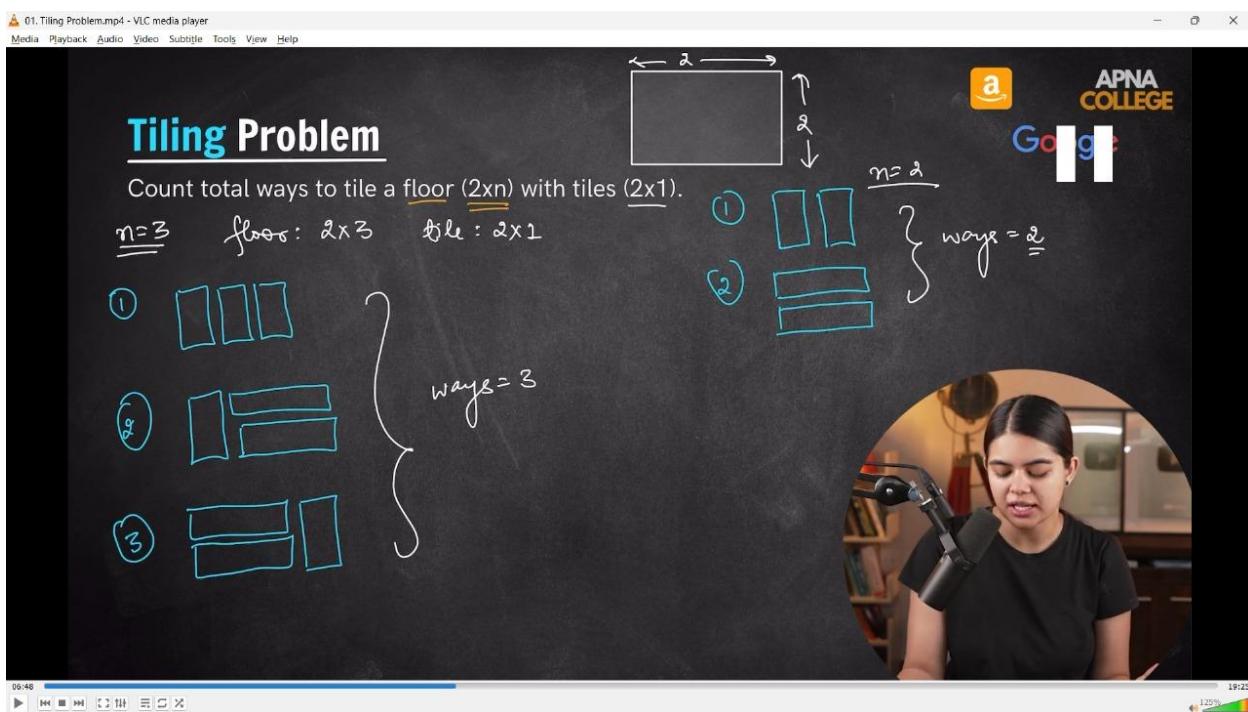
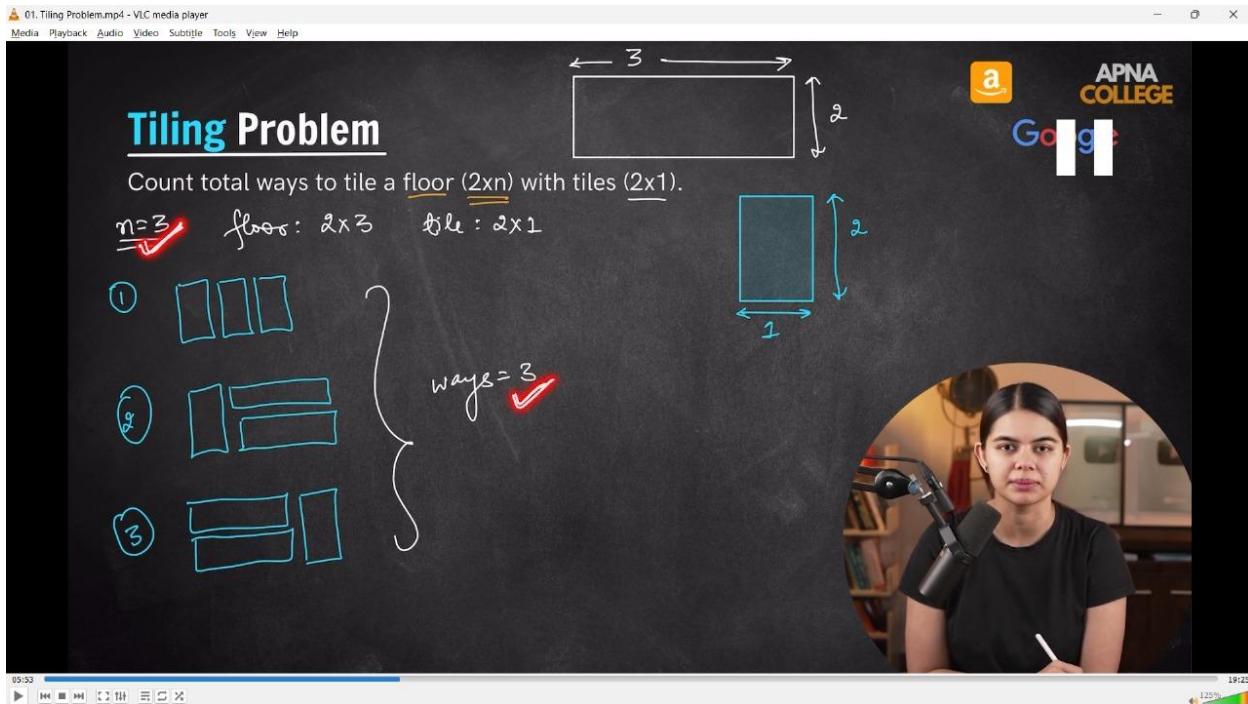
```

// _____

```

### //3) Tiling problem





01.Tiling Problem.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Tiling Problem

choice  $[2 \times n]$

vertical  $[2 \times (n-1)]$

horizontal

① kaam  
 ② recursive call  
 ③ Base case



APNA COLLEGE

Google



10:46 19:25 125%

01.Tiling Problem.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Tiling Problem

Count total ways to tile a floor  $(2 \times n)$  with tiles  $(2 \times 1)$ .

$n=3$  floor:  $2 \times 3$  tile:  $2 \times 1$

①  ways = 3

②  ways = 2

③  ways = 1

$n=2$  ways = 2



13:41 19:25 125%

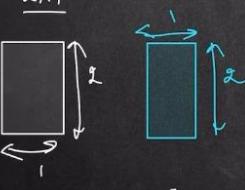
01.Tiling Problem.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Tiling Problem

$2 \times n$

$m=0 \quad 2 \times 0$   
 $\boxed{1 \quad \cancel{\text{ways} = 1}}$

$m=1 \quad 2 \times 1$   
  
 $\cancel{\text{ways} = 2}$

$m=2 \quad \cancel{\text{ways} = 2}$

$m=3 \quad \cancel{\text{ways} = 3}$

$m=4 \quad \cancel{\text{ways} = 5}$

① kaam ✓  
② recursive call ✓  
③ Base case ✓

recurrence relation

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

APNA COLLEGE

Google

tp(int n){  
 // vertical  
 ans1 ← tp(n-1)  
 // horizontal  
 ans2 ← tp(n-2)  
 ans1 + ans2  
}

19:18 19:25 125%



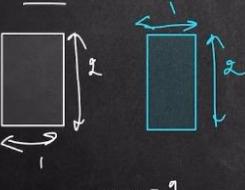
01.Tiling Problem.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Tiling Problem

$2 \times n$

$m=0 \quad 2 \times 0$   
 $\boxed{1 \quad \cancel{\text{ways} = 1}}$

$m=1 \quad 2 \times 1$   
  
 $\cancel{\text{ways} = 2}$

① kaam ✓  
② recursive call ✓  
③ Base case ✓

recurrence relation

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

APNA COLLEGE

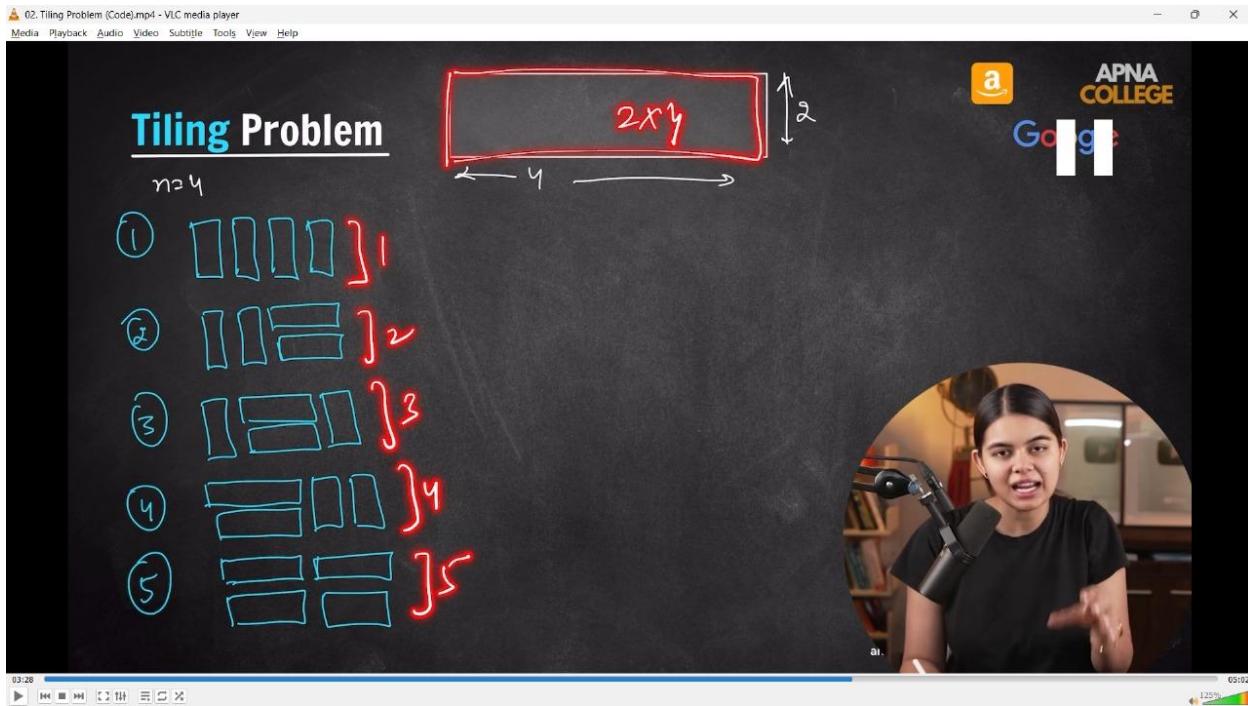
Google

$f(2) = \cancel{f(1)} + \cancel{f(0)}$

tp(int n){  
 // vertical  
 ans1 ← tp(n-1)  
 // horizontal  
 ans2 ← tp(n-2)  
 ans1 + ans2  
}

19:23 19:25 125%





```
// int tilingProblem(int n) // 2*n

// {
//   if (n == 0 || n == 1)
//   {
//     return 1;
//   }

//   // vertical
//   int ans1 = tilingProblem(n - 1); // 2*n-1

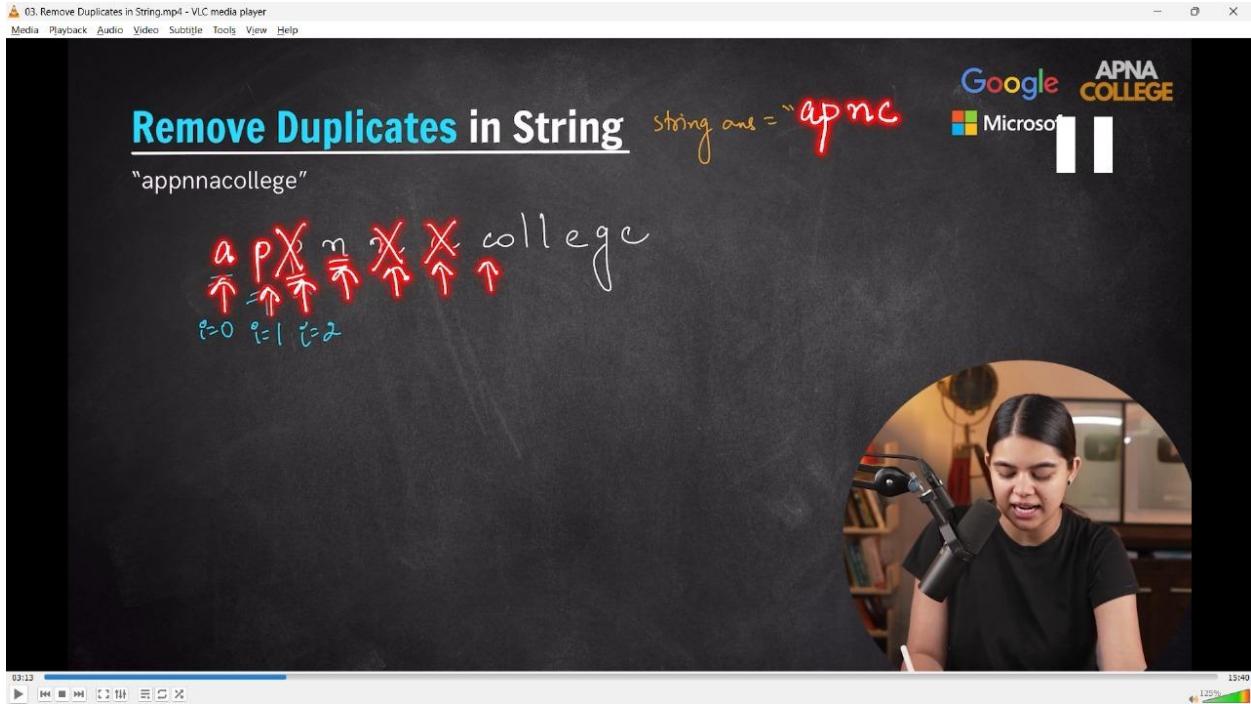
//   // horizontal
//   int ans2 = tilingProblem(n - 2); // 2*n-2

//   int ans = ans1 + ans2; //Or we could directly in a single line as - return ans = tilingProblem(n-1)+tilingProblem(n-2)
```

```
// }

// int main()
// {
//   cout << tilingProblem(2) << endl; // 2
//   cout << tilingProblem(3) << endl; // 3
//   cout << tilingProblem(4) << endl; // 5
//   cout << tilingProblem(5) << endl; // 8
// }
// 
```

#### 4) Remove Duplicates from the String –



03. Remove Duplicates in String.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Remove Duplicates in String

string ans = apnacoleg

"appnnacollege" ← str

bool map[26] [FT|F|TF|F|TF|TF|TF|F|F]  
 ↴  
 'a' 0  
 'b' 1  
 'c' 2  
 'd' 3  
 'e' 4  
 'f' 5  
 'g' 6  
 'h' 7  
 'i' 8  
 'j' 9  
 'k' 10  
 'l' 11  
 'm' 12  
 'n' 13  
 'o' 14  
 'p' 15  
 'q' 16  
 'r' 17  
 's' 18  
 't' 19  
 'u' 20  
 'v' 21  
 'w' 22

removeDlp(str, ans, i, map[ ])

index 0 =

① kaam ✓  
 ② recursive fnx  
 ③ Base Case

12:01 12:40 125% 12:40

03. Remove Duplicates in String.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Remove Duplicates in String

string ans = apnacoleg

"appnnacollege" ← str

map[i]

$f(str, ans, i+1, map)$

$(str, ans + str[i], i+1, map)$

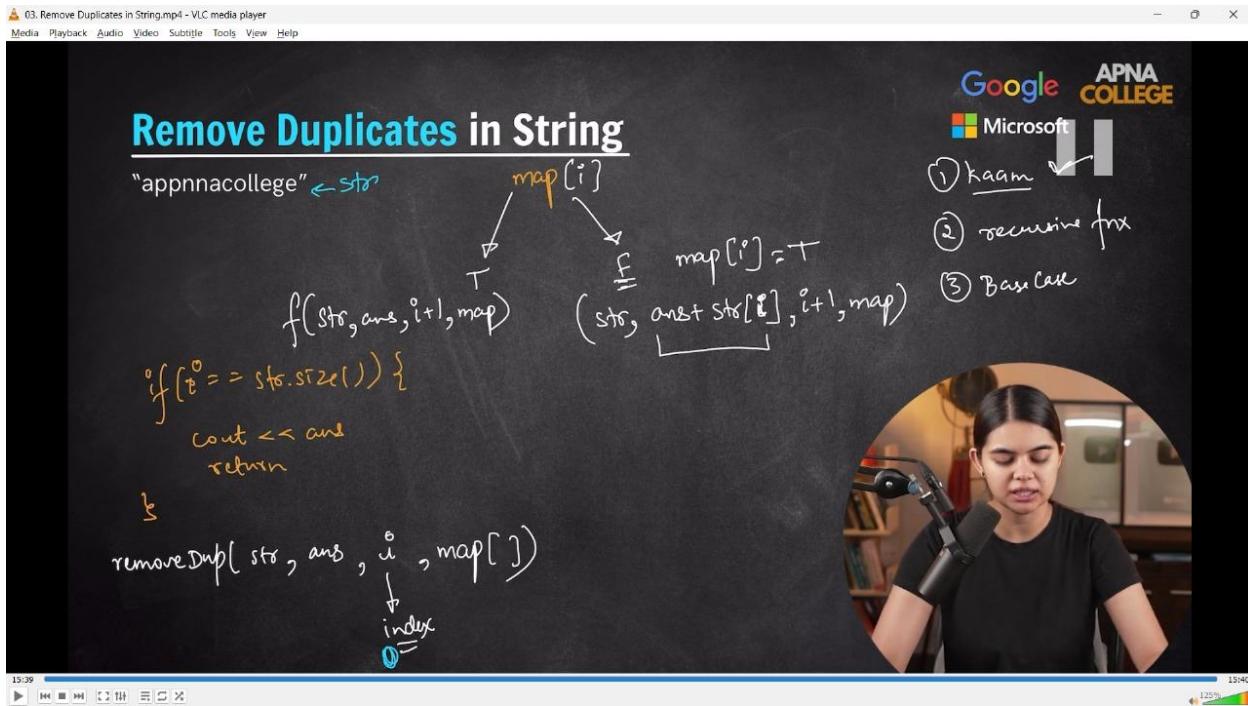
if ( $t == str[i]$ ) {  
 cout << ans  
 return  
}  

removeDlp(str, ans, i, map[ ])

index 0 =

① kaam ✓  
 ② recursive fnx  
 ③ Base Case

15:27 15:40 125% 15:40



```
// void removeDuplicates(string str, string ans, int i, int map[26])
// {
//     if (i == str.size())
//     {
//         cout << "ans : " << ans << endl;
//         return;
//     }
//
//     char ch = str[i];
//     int mapIdx = (int)(ch - 'a');
//
//     if (map[mapIdx] == true) // duplicate
//     {
//         removeDuplicates(str, ans, i + 1, map);
//     }
// }
```

```
// else // duplicate
// {
//     map[mapIdx] = true;
//     removeDuplicates(str, ans + str[i], i + 1, map);
// }
// }

// int main()
//{
//     string str = "Miicrosoftt Hyderabaaaad";
//     string ans = "";
//     int map[26] = {false};

//     removeDuplicates(str, ans, 0, map); //ans : Microsft Hydeab
// }

// _____
//5) Friends Pairing Problem -
```

05. Friends Pairing Problem.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Friends Pairing Problem

Find total ways in which n friends can be paired up.

Each friend can only be paired once.

$n=3$

pair

05. Friends Pairing Problem.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Friends Pairing Problem

Find total ways in which n friends can be paired up.

Each friend can only be paired once.

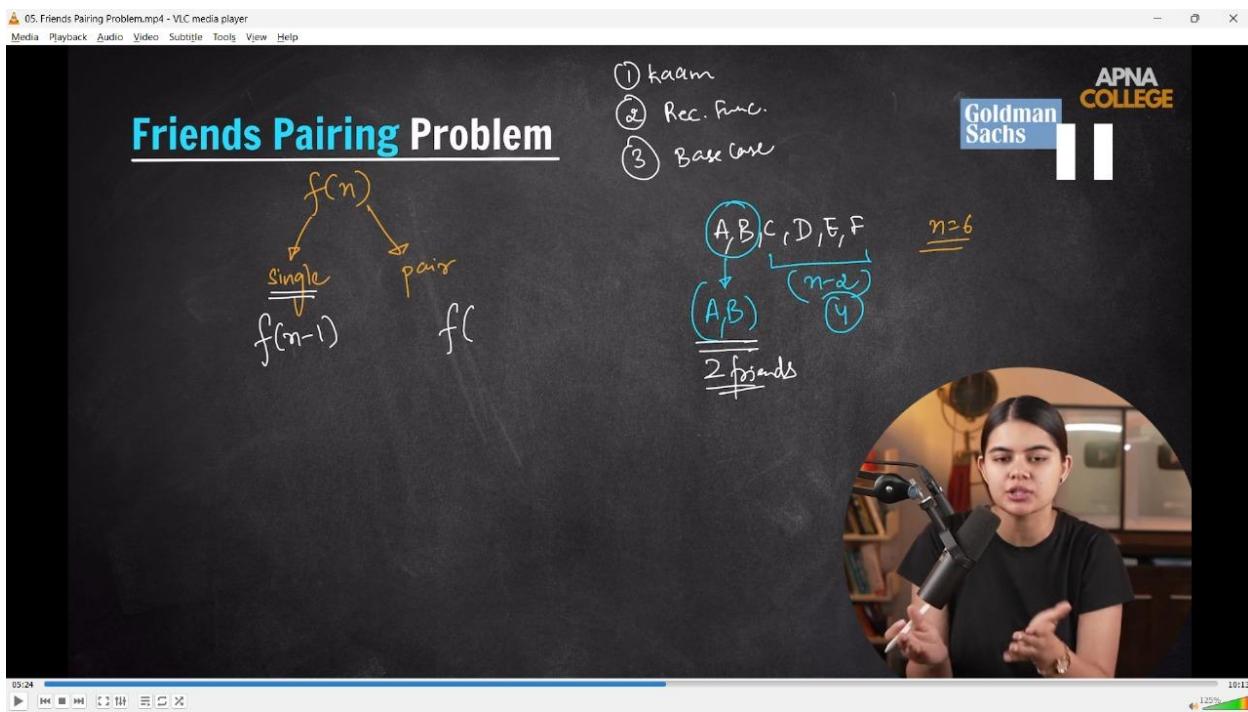
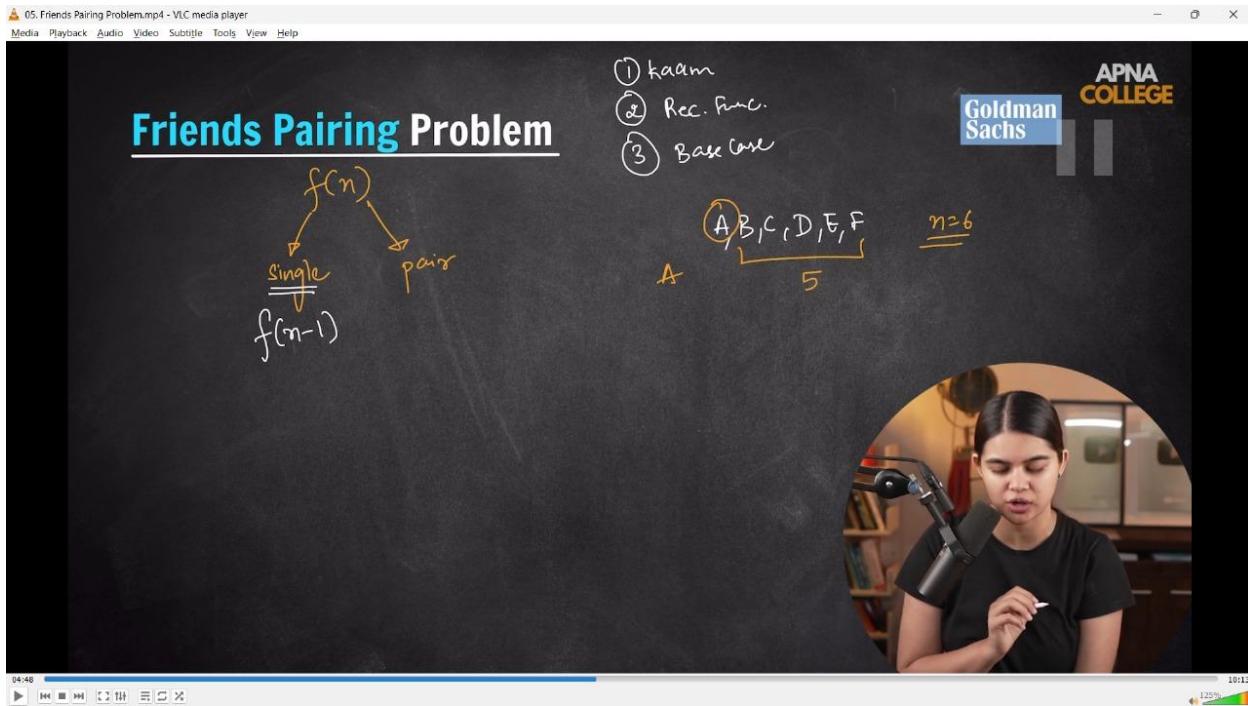
$n=1$       ways = 1

$n=2$       ways = 2

$n=3$       ways = 3

Permutation & Combination      P & C

$n=4, 5 \dots$



05. Friends Pairing Problem.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Friends Pairing Problem

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

recurrence relation ↑

$A, \underline{B}, \underline{C}, \underline{D}, \underline{E}, F$        $\underline{\underline{n=6}}$

$AB$   
 $AC$   
 $AD$   
 $AE$   
 $AF$

10:04 10:13 125%

APNA COLLEGE  
Goldman Sachs

05. Friends Pairing Problem.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Friends Pairing Problem

$n=1$     ways = 1  
 $n=2$     ways = 2

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

recurrence relation ↑

10:11 10:13 125%

APNA COLLEGE  
Goldman Sachs

```
// int FriendPairing(int n)
//{
//    if (n == 1 || n == 2)
//    {

```

```

//      return n;

// }

// return FriendPairing(n - 1) + (n - 1) * FriendPairing(n - 2);

// }

// int main()

// {

//     cout << FriendPairing(3) << endl;//4

//     cout << FriendPairing(4) << endl;//10

//     cout << FriendPairing(5) << endl;//26

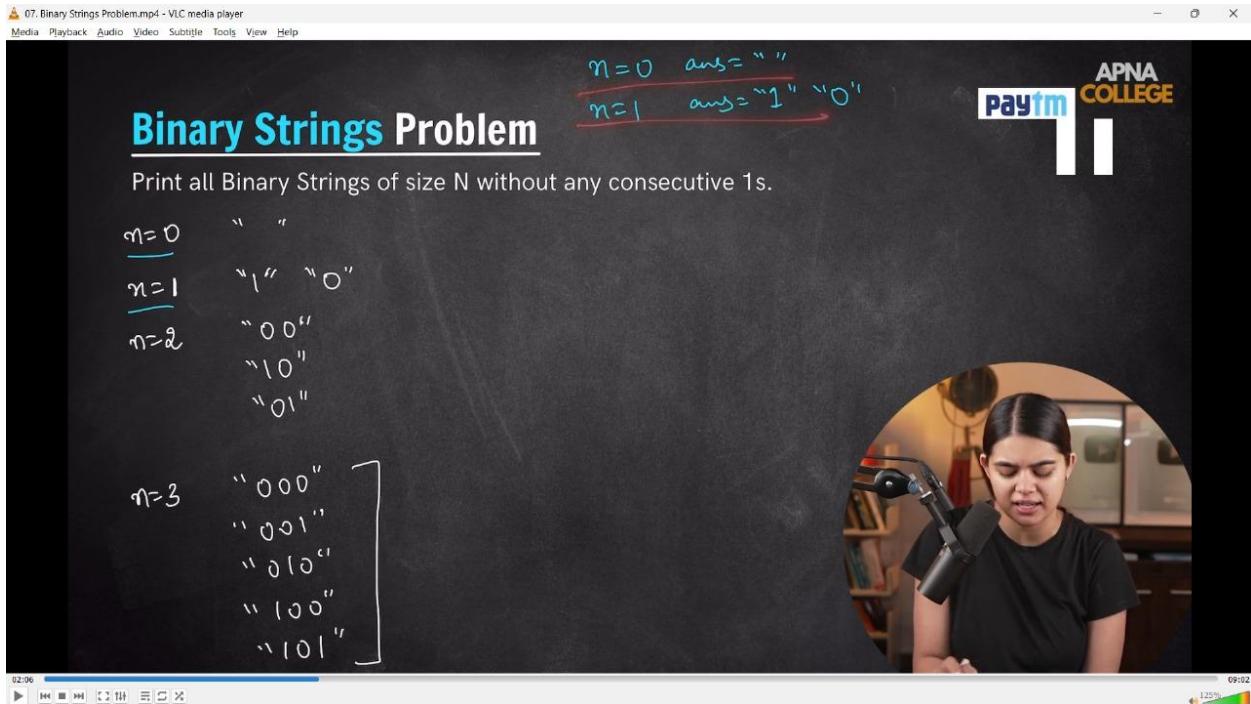
//     cout << FriendPairing(6) << endl;//76

// }

// _____

```

## //6) Binary String Problem -



07. Binary Strings Problem.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Binary Strings Problem

Print all Binary Strings of size  $\underline{N}$  without any consecutive 1s.

$n=3$

$1 \quad \underline{\quad} \quad \underline{\quad}$   
 $n=2$

$0 \quad \underline{0/1} \quad -$

$1 \quad \underline{0} \quad -$

$0 \quad \underline{\quad}$   
 $\underline{0/1} \quad f(n-1)$

$\underline{1} \quad \underline{\quad}$   
 $\underline{0} \quad f(n-1)$

$n=0 \quad ans = ""$   
 $n=1 \quad ans = "1" "0"$

(1) kaam ✓  
(2) recursive call ✓  
(3) BC ✓

06:15 09:02  
125%

07. Binary Strings Problem.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Binary Strings Problem

Print all Binary Strings of size  $\underline{N}$  without any consecutive 1s.

$F(n, \underline{lastPlace}, \underline{ans}) \{$

$\quad \underline{if (n == 0)}$   $\leftarrow$  prints ans

$\quad \underline{if (lastPlace == 1)}$

$\quad \quad F(n-1, 0, ans + '0') \checkmark$

$\quad \quad F(n-1, 1, ans + '1')$

$\quad } else \{$

$\quad \quad F(n-1, 0, ans + '0')$

$\}$

(1) kaam ✓  
(2) recursive call ✓  
(3) BC ✓

08:42 09:02  
125%

09. Binary Strings (Dry Run).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Binary Strings Problem

Dry Run for  $\underline{n = 3}$

Initial state:  $n=3, ans = "", lp = 0$

Recursion tree:

- $n=2, ans = 0, lp = 0$  →
  - $n=1, ans = 00, lp = 0$  →
    - $n=0, ans = 000, lp = 0$  (1)
    - $n=0, ans = 001, lp = 1$  (2)
  - $n=1, ans = 01, lp = 1$  (3)
- $n=2, ans = 1, lp = 1$  (4)

Final states (leaf nodes): (1) 000, (2) 001, (3) 010, (4) 10, (5) 101.

APNA COLLEGE Paytm

09. Binary Strings (Dry Run).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Binary Strings Problem

Dry Run for  $\underline{n = 3}$

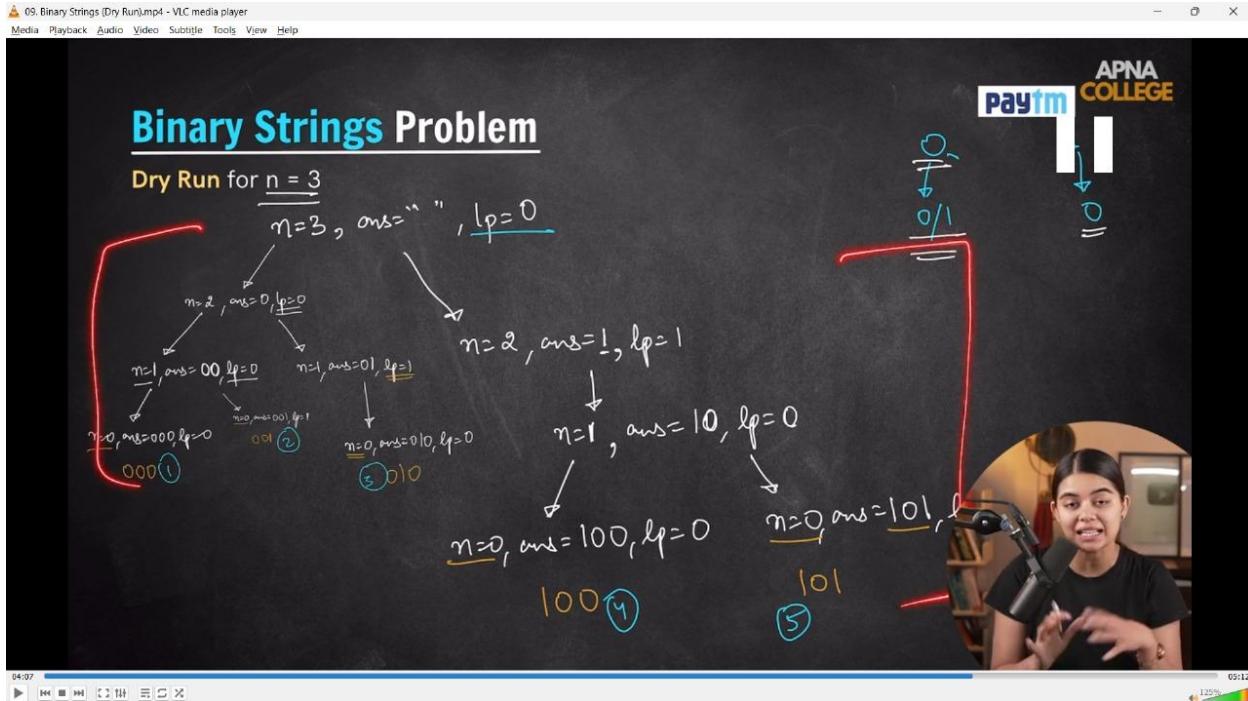
Initial state:  $n=3, ans = "", lp = 0$

Recursion tree:

- $n=2, ans = 0, lp = 0$  →
  - $n=1, ans = 00, lp = 0$  →
    - $n=0, ans = 000, lp = 0$  (1)
    - $n=0, ans = 001, lp = 1$  (2)
  - $n=1, ans = 01, lp = 1$  (3)
- $n=2, ans = 1, lp = 1$  (4)

Final states (leaf nodes): (1) 000, (2) 001, (3) 010, (4) 10, (5) 101.

APNA COLLEGE Paytm



```
// void binaryString(int n, int lastPlace, string ans)

// {
//   if (n == 0)
//   {
//     cout << ans << endl;
//     return;
//   }
//   if (lastPlace != 1)
//   {
//     binaryString(n - 1, 0, ans + '0');
//     binaryString(n - 1, 1, ans + '1');
//   }
//   else
//   {
//     binaryString(n - 1, 0, ans + '0');
```

```
// }

// }

// int main()

// {

//     string ans = "";

//     binaryString(2, 0, ans);

//     /*

//     00

//     01

//     10

//     */

//     cout<<endl;

//     binaryString(3, 0, ans);

//     /*

//     000

//     001

//     010

//     100

//     101

//     */

//     cout << endl;

//     binaryString(4, 0, ans);

//     /*

//     0000

//     0001

//     0010
```

```
// 0100
// 0101
// 1000
// 1001
// 1010
// */
// cout << endl;
// binaryString(5, 0, ans);
// /*
// 00000
// 00001
// 00010
// 00100
// 00101
// 01000
// 01001
// 01010
// 10000
// 10001
// 10010
// 10100
// 10101

// */
// cout << endl;
// }
```

// \_\_\_\_\_

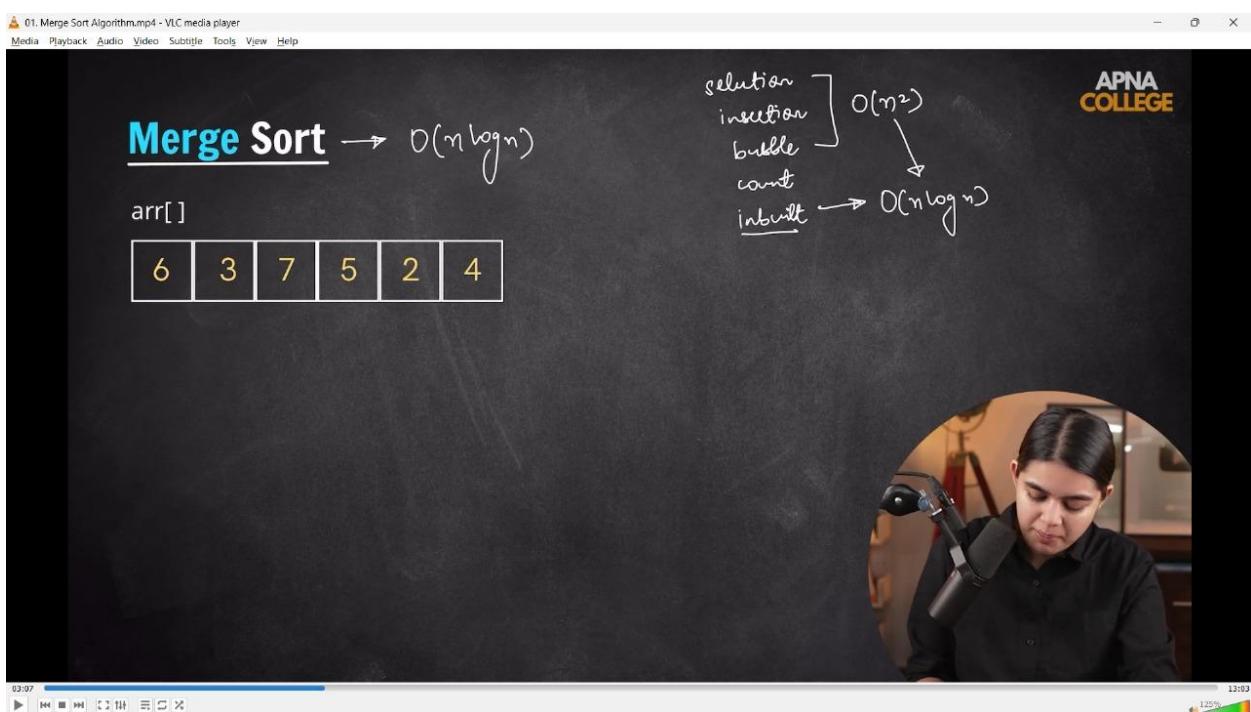
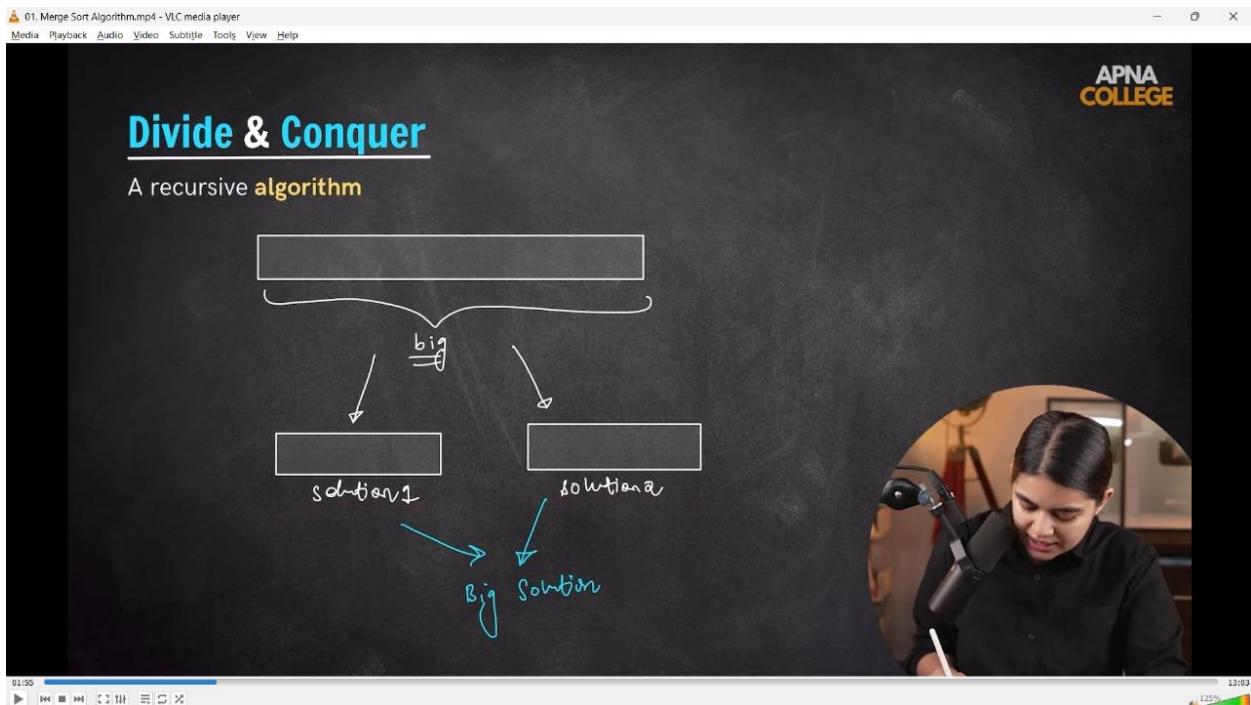
\_\_\_\_\_

**[10] Divide & Conquer –**

- 1) Merge Sort Algo -
- 2) Quick Sort Algorithm -
  - 2.1) Quick Sort Worst Case Scenario -
- 3) Searching In Rotated Array Problem -

## 10) Divide & Conquer –

### //1) Merge Sort Algo -



01. Merge Sort Algorithm.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Merge Sort

Approach

- ① Divide
 
$$\text{mid} = \frac{s_i + e_i}{2}$$
- ② mergeSort(left)  $\cup$  mergeSort(right)
- ③ merge( )

$s_i = 0$     $e_i = n-1 = 5$

$m_id = \frac{(0+5)}{2} = 2.5$

$L_{mid} = \frac{(0+2)}{2} = 1$

$R_{mid} = \frac{(3+5)}{2} = 4$

$s_i = e_i$

Base Case

APNA COLLEGE

12:37 12:03 125%

02. Merge \_ Combine Step.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Merge Sort

Merge / Combine Step

$i = s_i$

$j = m_id + 1$

$s_i = 0$     $e_i = 5$

$m_id = \frac{(0+5)}{2} = 2.5$

$i = 1$

$j = 3$

$m_id + 1 = 3$

$s_i = 0$     $e_i = 2$

$m_id = \frac{(0+2)}{2} = 1$

$i = 1$

$s_i = 3$     $e_i = 5$

$m_id = \frac{(3+5)}{2} = 4$

$j = 3$

$m_id + 1 = 4$

APNA COLLEGE

06:14 12:18 125%

02. Merge \_ Combine Step.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Merge Sort

Merge / Combine Step

$i = \&i^o$   
 $j = mid + 1$

APNA COLLEGE

6 | 3 | 7 | 5 | 2 | 4  
mid  
[ 6, 3, 7 ] [ 5, 2, 4 ]  
ms  
[ 3 | 6 | 7 ] mid i  
[ 4 | 5 ] ei  
temp (vec)  
✓ ✓ ✓ ✓ ✓ ✓

06:45 12:18 125%

02. Merge \_ Combine Step.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Merge Sort

Merge / Combine Step

```
ms(aar, si, ei) {
    if(si >= ei)
        return
    mid = (si+ei)/2
    ms(aar, si, mid) //left ✓
    ms(aar, mid+1, ei) //right ✓
    merge(aar, si, ei, mid)
}
```

APNA COLLEGE

6 | 3 | 7 | 5 | 2 | 4  
merge(aar, si, ei, mid) {  
vector<int>  
i = &i^o → left  
j = mid + 1 → right  
while (i <= mid && j <= ei) {  
aar[i], aar[j] → vectors  
[ ] ✓ ✓

11:51 12:18 125%

02. Merge \_Combine Step.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Merge Sort

### Merge / Combine Step

```

ms(aar, si, ei) {
    if(si >= ei)
        return
    mid = (si+ei)/2
    ms(aar, si, mid) || left ✓
    ms(aar, mid+1, ei) || right ✓
    c(aar, si, ei, mid)
}

```

APNA COLLEGE

merge(aar, si, ei, mid) {  
 vector<int>  
 i = si → left  
 j = mid+1 → right  
 while (i <= mid && j <= ei) {  
 aar[i], aar[j] → vectors  
 }
}

copy vector to original array

12:14 12:18 125%

04. Merge Sort (Summary).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Merge Sort

### summary

APNA COLLEGE

mid

s=0 e=5

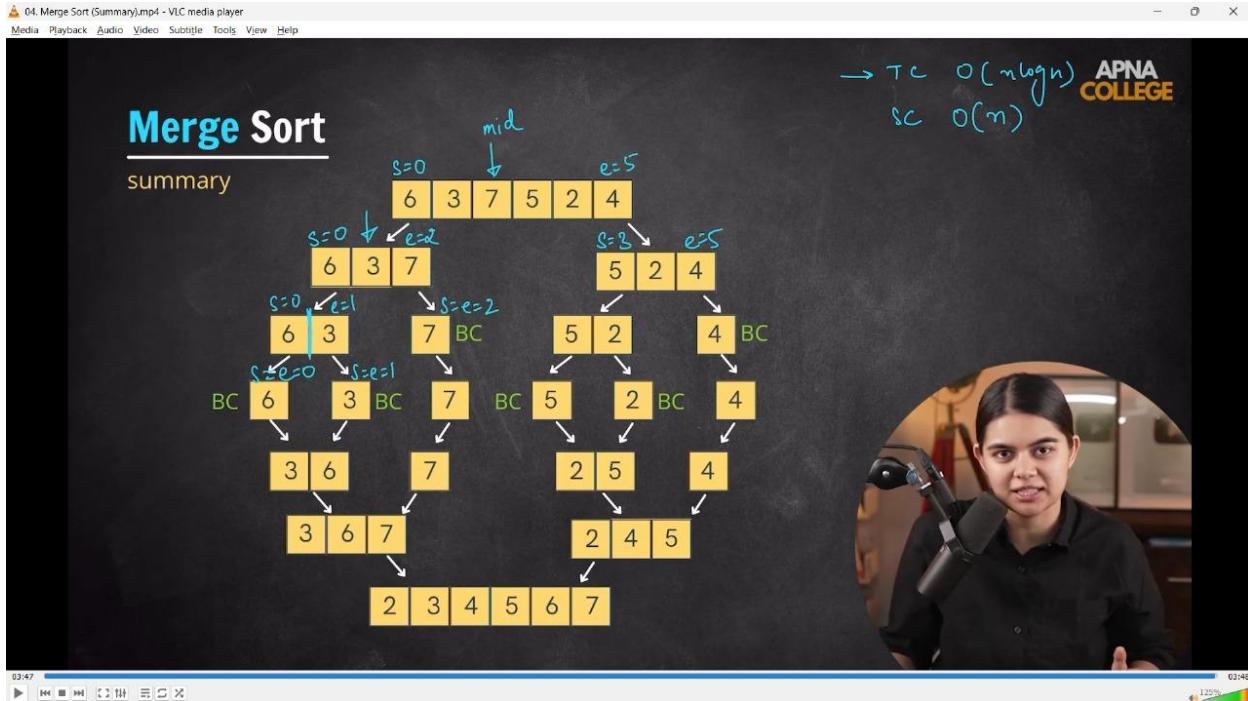
s=0 e=2

s=0 e=1

BC

3 6 7 2 5 4

03:19 02:48 125%



```
void Merge(int arr[], int si, int mid, int ei) // O(n)
```

```
{
```

```
vector<int> temp;
```

```
int i = si;
```

```
int j = mid + 1;
```

```
while (i <= mid && j <= ei)
```

```
{
```

```
if (arr[i] <= arr[j])
```

```
{
```

```
temp.push_back(arr[i]);
```

```
i++;
```

```
}
```

```
else
```

```
{
```

```

        temp.push_back(arr[j]);
        j++;
    }
}

while (i <= mid)
{
    temp.push_back(arr[i]);
    i++;
}

while (j <= ei)
{
    temp.push_back(arr[j]);
    j++;
}

// Converting vector to an aOriginal Array
for (int i = si, x = 0; i <= ei; i++)
{
    arr[i] = temp[x++];
}

void mergeSort(int arr[], int si, int ei) // O(log n), so total complexity - O(nlogn)
{
    if (si >= ei)

```

```
{  
    return;  
}  
  
int mid = si + (ei - si) / 2;  
  
mergeSort(arr, si, mid); // LeftHalf  
mergeSort(arr, mid + 1, ei); // RightHalf  
Merge(arr, si, mid, ei); // Conquer  
}
```

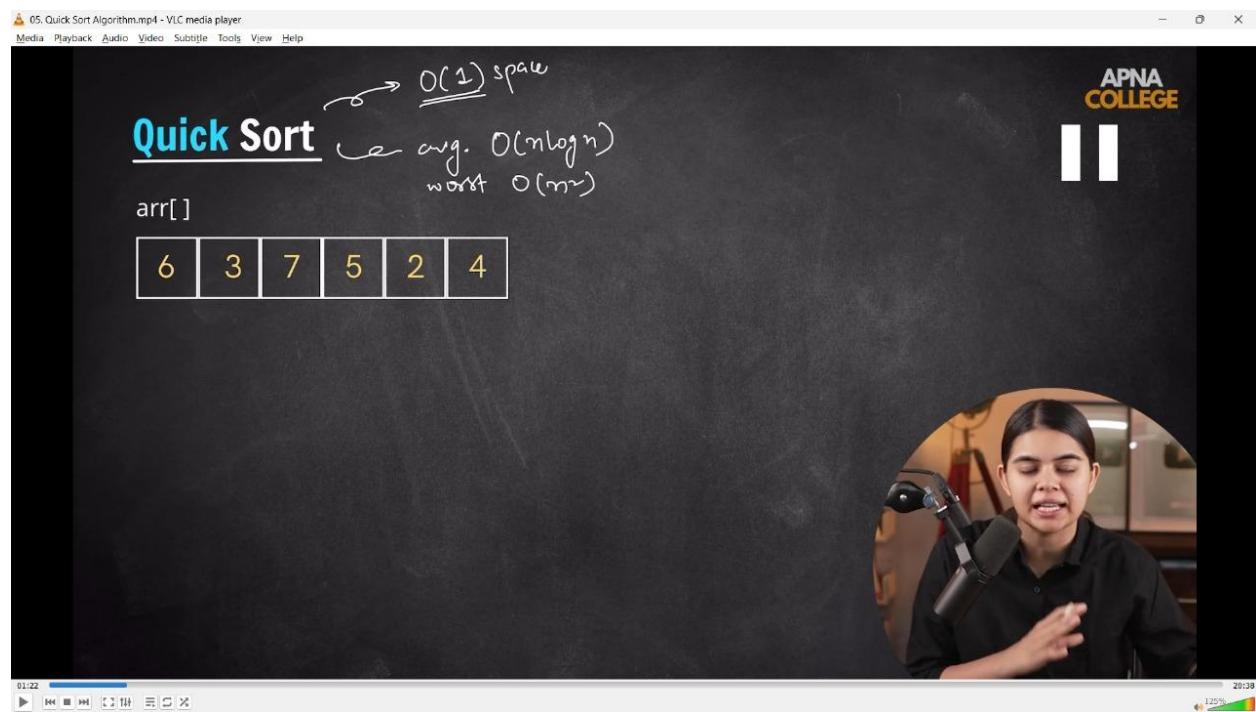
```
void printArray(int arr[], int n)  
{  
    cout << "Sorted Array is - " << endl;  
    for (int i = 0; i < n; i++)  
    {  
        cout << arr[i] << " ";  
    }  
    cout << endl;  
}  
  
int main()  
{  
    int arr[6] = {6, 3, 7, 5, 2, 4};  
    int n = 6;  
  
    mergeSort(arr, 0, n - 1);  
    printArray(arr, n);
```

```
/*
Sorted Array is -
```

```
2 3 4 5 6 7
```

```
*/  
}  
  
// _____
```

## //2) Quick Sort Algorithm -



05. Quick Sort Algorithm.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Pivot & Partition Approach

$\downarrow$  special idx  $\rightarrow$  ending idx

① pivot  
② partition  
③ QS (left) QS (right)

$[2, 3] \leftarrow l$   $[5, 6, 7] \rightarrow r$

APNA COLLEGE

07:36 20:38 125%

05. Quick Sort Algorithm.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Pivot & Partition Approach

$s^i, p^{i-1}, e^i$

$p^{i+1}$

$s^i = \text{partition}(arr, s_i, e_i)$

$p^{i+1} = \text{partition}(arr, s_i, e_i)$

$QS(s^i, p^{i-1})$

$QS(p^{i+1}, e^i)$

① pivot  
② partition  
③ QS (left) QS (right)

$[2, 3] \leftarrow l$   $[5, 6, 7] \rightarrow r$

APNA COLLEGE

10:38 20:38 125%

05. Quick Sort Algorithm.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Partition Step

$$i = i - 1$$

less

less = 4

greater

original

13:43 20:38 125%

05. Quick Sort Algorithm.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Partition Step

$$i = i - 1$$

less

less = 4

greater

i++;

swap(arr[i], arr[j])

original

14:46 20:38 125%

05. Quick Sort Algorithm.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Partition Step

$$i = i-1$$

less  $\leq$

$$i++;$$

$$\text{swap}(a[i], a[j])$$

APNA COLLEGE

original

3 | 6 | 7 | 5 | 2 | 4

i = -1      i = 0      i = 1      i = 2      i = 3      i = 4      i = 5

pivot = arr[i]

less

greater

15:09 20:38

05. Quick Sort Algorithm.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Partition Step

$$i = i-1$$

less  $\leq$

$$i++;$$

$$\text{swap}(a[i], a[j])$$

$$arr[i]$$

APNA COLLEGE

original

3 | 2 | 7 | 5 | 6 | 4

i = -1      i = 0      i = 1      i = 2      i = 3      i = 4      i = 5

pivot = arr[i]

less

greater

17:36 20:38

05. Quick Sort Algorithm.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Partition Step

$$i = ei - 1$$

$$\downarrow$$

less des

$$i++;$$

$$\text{swap}(arr[i], arr[j])$$

$$\cancel{\text{pivotIdx}}$$

$$i = \cancel{ei}$$

APNA COLLEGE

original

3 | 2 | 7 | 5 | 6 | 4

i = -1      i = 0      i = 1      i = 2

05. Quick Sort Algorithm.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Partition Step

```
int partition(arr, si, ei) {
    i = ei - 1; pivot = arr[ei]
    for(j = si; j < ei; j++) {
        if(arr[j] <= pivot) {
            i++;
            swap(arr[i], arr[j])
        }
    }
    i++;
    swap(arr[i], arr[ei])
    return
}
```

APNA COLLEGE

original

3 | 2 | 4 | 5 | 6 | 7

i = -1      i = 0      i = 1      i = 2

```
int partition(int arr[], int si, int ei)
```

```
{
```

```
    int i = si - 1;
```

```
    int pivot = arr[ei];
```

```
for (int j = si; j < ei; j++)  
{  
    if (arr[j] <= pivot)  
    {  
        i++;  
        swap(arr[i], arr[j]);  
    }  
}  
i++;  
swap(arr[i], arr[ei]);  
// pivotindex = i  
return i;  
}
```

```
void QuickSort(int arr[], int si, int ei)//O(n*logn)
```

```
{  
    if (si >= ei)  
    {  
        return;  
    }
```

```
    int pivotIdx = partition(arr, si, ei);
```

```
    QuickSort(arr, si, pivotIdx - 1); // Left Half
```

```
    QuickSort(arr, pivotIdx + 1, ei); // Right Half
```

```
}
```

```
void printArr(int arr[], int n)
```

```
{
```

```
    for (int i = 0; i < n; i++)
```

```
    {
```

```
        cout << arr[i] << " ";
```

```
    }
```

```
    cout << endl;
```

```
}
```

```
int main()
```

```
{
```

```
    int arr[6] = {6, 3, 7, 5, 2, 4};
```

```
    int n = 6;
```

```
    QuickSort(arr, 0, n - 1);
```

```
    printArr(arr, n); // 2 3 4 5 6 7
```

```
}
```

```
// 2.1) Quick Sort Worst Case Scenario
```

07. Worst Case Time Complexity - Quick Sort.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

**Quick Sort**

**Worst Case** Time Complexity =  $O(N^2)$

partition ✓  
Loop

$$\frac{n \cdot (n+1)}{2} \approx O(n^2)$$

\*This complexity occurs when repeatedly the smallest or the largest element of the array becomes the pivot.

APNA COLLEGE

04:17 03:21 125%

/\* In the Worst Case of Quick Sort, the complexity will be  $O(n^2)$  and the Worst case occurs when either in INCREASING Order Or in DECREASING Order

So, the Pivot element will be at last , so there will be an AP

\*/

```
int partition(int arr[], int si, int ei)
```

```
{
```

```
    int i = si - 1;
```

```
    int pivot = arr[ei];
```

```
    for (int j = si; j < ei; j++)
```

```
{
```

```
        if (arr[j] <= pivot)
```

```
{
```

```
            i++;
```

```
            swap(arr[i], arr[j]);
```

```
    }

}

i++;

swap(arr[i], arr[ei]);

// pivotindex = i

return i;

}

void QuickSort(int arr[], int si, int ei) // O(n*logn)

{

if (si >= ei)

{

return;

}

int pivotIdx = partition(arr, si, ei);

QuickSort(arr, si, pivotIdx - 1); // Left Half

QuickSort(arr, pivotIdx + 1, ei); // Right Half

}

void printArr(int arr[], int n)

{

for (int i = 0; i < n; i++)

{

cout << arr[i] << " ";
```

```

    }

    cout << endl;

}

int main()
{
    int arr[6] = {1, 2, 3, 4, 5, 6};

    int n = 6;

    QuickSort(arr, 0, n - 1);

    printArr(arr, n); // 1 2 3 4 5 6
}

```

// \_\_\_\_\_

### //3) Searching In Rotated Array Problem

The screenshot shows a VLC media player window with the following details:

- Title Bar:** 08. Search in Rotated Sorted Array.mp4 - VLC media player
- Menu Bar:** Media, Playback, Audio, Video, Subtitle, Tools, View, Help
- Video Content:**
  - Chalkboard Notes:**
    - Search in Rotated Sorted**
    - input : Rotated, sorted array with distinct nums ; ascending order
    - Find index of target = 0
    - arr**
    - A diagram of a rotated sorted array with 8 cells. The first 4 cells contain 4, 5, 6, 7 respectively. The last 4 cells contain 0, 1, 2, 3. The cell containing 0 is circled in blue.
    - ans = 4
    - linear search  $\rightarrow O(n)$**
    - $O(\log n)$
  - APNA COLLEGE** logo in the top right corner of the chalkboard.
  - Microphone Feed:** A circular video feed in the bottom right corner shows a person with dark hair speaking into a microphone.

06. Search in Rotated Sorted Array.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

$\text{tar} = 0$

## Search in Rotated Sorted

Modified Binary Search Approach

$\text{arr}[mid] == \text{tar}$   
return mid;

APNA COLLEGE

06:09 18:08 125%

06. Search in Rotated Sorted Array.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

$\text{tar} = 0$

## Search in Rotated Sorted

Modified Binary Search Approach

$\text{arr}[mid] == \text{tar}$   
return mid;

case 1:  
 $L_1$

APNA COLLEGE

09:00 18:08 125%

06. Search in Rotated Sorted Array.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Search in Rotated Sorted

Modified Binary Search Approach

$\text{tar} = 0$

Diagram showing a rotated sorted array divided into two parts, L1 and L2, with mid pointing to 7.

Case 1:

- case a:  $\text{arr}[si] \leq \text{tar} < \text{arr}[mid]$  // left  $f(si, mid-1)$
- case b: false // right  $f(mid+1, ei)$

Condition ①:  $(\text{arr}[mid] == \text{tar})$  return mid;

12:35 18:58 125%

06. Search in Rotated Sorted Array.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Search in Rotated Sorted

Modified Binary Search Approach

$\text{tar} = 0$

Diagram showing a rotated sorted array divided into two parts, L1 and L2, with mid pointing to 7.

Case 1:

- case a:  $\text{arr}[si] \leq \text{tar} < \text{arr}[mid]$  // left  $f(si, mid-1)$
- case b: false // right  $f(mid+1, ei)$

Condition ①:  $(\text{arr}[mid] == \text{tar})$  return mid;

Case 2:

L2

13:50 18:58 125%

06. Search in Rotated Sorted Array.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Search in Rotated Sorted

Modified Binary Search Approach

$\text{tar} = 0$

$\text{arr}[si]$   $\text{arr}[ei]$

$\text{L1}$   $\text{L2}$   $\text{mid}$

$a$   $b$   $c$   $d$

$\text{Case 1:}$   $\text{L1}$  case a:  $\text{arr}[si] \leq \text{tar} \leq \text{arr}[mid]$   
 $\text{L2}$  case b: false  $\text{L1}$   $\text{L2}$   $\text{mid}$

$\text{Case 2:}$   $\text{L2}$  case c:  $\text{arr}[mid] \leq \text{tar} \leq \text{arr}[ei]$   
 $\text{L1}$  case d: else  $\text{L1}$   $\text{L2}$   $\text{mid}$

①  $(\text{arr}[mid] == \text{tar})$   
 $\text{return mid;}$

APNA COLLEGE

18:03 18:58 125%

A woman is speaking into a microphone while writing on the chalkboard.

06. Search in Rotated Sorted Array.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Search in Rotated Sorted

Modified Binary Search Approach

$\text{tar} = 0$

$\text{arr}[si]$   $\text{arr}[ei]$

$\text{L1}$   $\text{L2}$   $\text{mid}$

$a$   $b$   $c$   $d$

$\text{Case 1:}$   $\text{L1}$  case a:  $\text{arr}[si] \leq \text{tar} \leq \text{arr}[mid]$   
 $\text{L2}$  case b: false  $\text{L1}$   $\text{L2}$   $\text{mid}$

$\text{Case 2:}$   $\text{L2}$  case c:  $\text{arr}[mid] \leq \text{tar} \leq \text{arr}[ei]$   
 $\text{L1}$  case d: else  $\text{L1}$   $\text{L2}$   $\text{mid}$

①  $(\text{arr}[mid] == \text{tar})$   
 $\text{return mid;}$

$\text{arr}[si] \leq \text{mid} \rightarrow \text{L1}$   
 $\text{F} \rightarrow \text{L2}$

APNA COLLEGE

18:52 18:58 125%

A woman is speaking into a microphone while writing on the chalkboard.

09. Rotated Sorted (Pseudocode).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Search in Rotated Sorted

Pseudocode

```

search( arr, si, ei, tar)
    mid = si + (ei - si)/2;
    if( arr[mid] == tar)
        return mid;
    if( arr[si] <= arr[mid]) { //L1
        if( arr[si] <= tar <= arr[mid])
            left half search(si, mid-1)
        else
            right half search(mid+1, ei)
    }

```

APNA COLLEGE

02:26 06:17 125%

09. Rotated Sorted (Pseudocode).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

## Search in Rotated Sorted

$O(\log n)$

Pseudocode

```

search( arr, si, ei, tar)
    mid = si + (ei - si)/2;
    if( arr[mid] == tar)
        return mid;
    if( arr[si] <= arr[mid]) { //L1
        if( arr[si] <= tar <= arr[mid])
            left half search(si, mid-1)
        else
            right half search(mid+1, ei)
    } else { //L2
        if( arr[mid] <= tar <= arr[ei])
            right search(mid+1, ei)
        else
            left search( si, mid-1)
    }

```

APNA COLLEGE

04:29 06:17 125%

09. Rotated Sorted (Pseudocode).mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Search in Rotated Sorted

$O(\log n)$

Pseudocode

```

search(arr, si, ei, tar)
    mid = si + (ei - si)/2;
    if (arr[mid] == tar)
        return mid;

    if (arr[si] <= arr[mid]) { //L1
        if (arr[si] <= tar <= arr[mid])
            left half search(si, mid-1)
        else
            right half search(mid+1, ei)
    } else { //L2
        if (arr[mid] <= tar <= arr[ei])
            right search(mid+1, ei)
        else
            left search(si, mid-1)
    }
}

```

if ( $si > ei$ )  
return -1;

```
int search(int arr[], int si, int ei, int target)
```

```
{
```

```
if (si > ei)
```

```
{
```

```
return -1;
```

```
}
```

```
int mid = si + (ei - si) / 2;
```

```
if (arr[mid] == target)
```

```
{
```

```
return mid;
```

```
}
```

```
if (arr[si] <= arr[mid]) // it's for L1
```

```
{
```

```
if (arr[si] <= target && target <= arr[mid])
{
    // For left Half
    return search(arr, mid + 1, ei, target);
}

else
{
    // For Right half
    return search(arr, mid + 1, ei, target);
}

}

else
{
    // Now it's for L2
    if (arr[mid] <= target && target <= arr[ei])
    {
        // For Right Half
        return search(arr, mid + 1, ei, target);
    }

    else
    {
        // For left half
        return search(arr, si, mid - 1, target);
    }
}
```

```
int main()
{
    int arr[7] = {4, 5, 6, 7, 0, 1, 2};

    int n = 7;

    cout << "idx : " << search(arr, 0, n - 1, 0) << endl; // idx : -1
    cout << "idx : " << search(arr, 0, n - 1, 2) << endl; // idx : 6
    cout << "idx : " << search(arr, 0, n - 1, 7) << endl; // idx : 3

    // T.C - O(logn)
}

// _____
```