Memory Address:

Array, Stack, String, Queue, Linked list

int a = 10;

a
10

address (hexadecimal)

it is an actual number.
e.g. 0x36a6
which tells us actually where our Variable is stored.

→ we can also print the Address
∴ Address of operator.
• operator we use to print the address of our variable is "&"

e.g:

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

int main(){

    int a = 10;

    cout << &a << endl;

    return 0;
}.
```

Output:
0x16d86b268

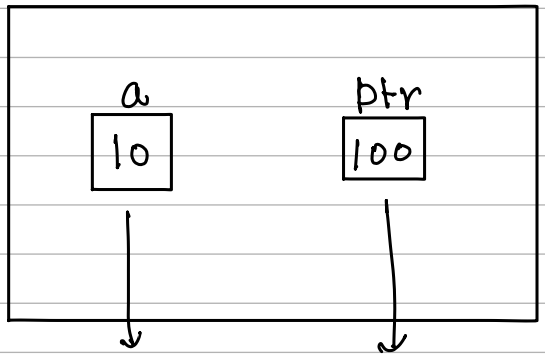
In this chapter we are going to Focus on:-

1. Pointers
2. multiple use of operators

e.g: "&" → Bit
"&" → Address.

* Pointers: Special variable that Stores address of other variable.

int a = 10;



- Syntax to create pointer.

→ Pointer is also an Variable.

```
int *ptr = &a
```

- `int` → datatype of Variable jiska address store karna hai.
- `*ptr` → name of Pointer.
- `&a` → "Showing address of a"

2. example:

```
Float x = 125.25;
Float *ptr = &x;
Cout << ptr << endl;
```

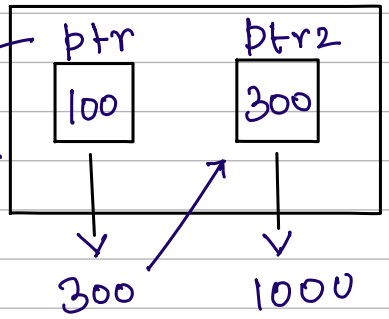
→ this gives us the address of our Variable "x".

`Cout << ptr<<endl;` = `Cout << &a << endl;`

* Pointer to Pointer: this means we make a pointer which stores address of other pointer.

`int** ptr2 = & ptr;` → Syntax

this is the pointer of a.



DEREFERENCE OPERATOR (*)

→ Dereference means getting the value Stored on Some Address.

- Print value by Pointer.

Value Stored at address "&a".

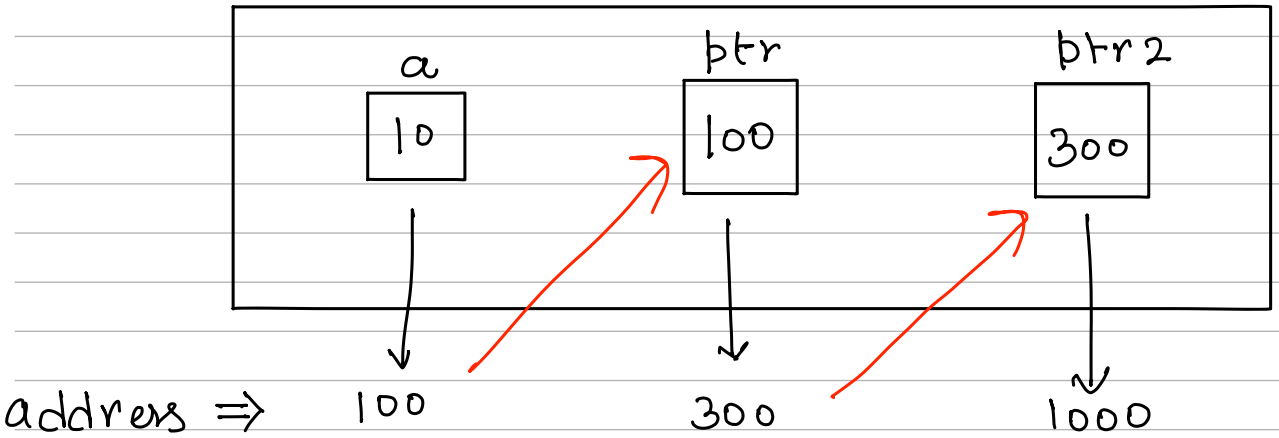
Syntax :→ `cout << *(&a) << endl;`

`*(&a)` → address of a.

eg:

```
int  a = 10;
int* ptr = &a;

cout << ptr << endl;

cout << *(ptr) << endl;
```

output.
0x16d86b268
→ 10



	CODE		output
here →	&a	–	100
	ptr	–	100
	&ptr	–	300
	ptr2	–	300
	&ptr2	–	1000
	*(&a)	–	10
	*(Ptr)	–	10
	*(Ptr2)	–	100

Syntax to initialize Dereference Variable.

```
int a = 10;
int* ptr = &a;

int** ptr2 = &ptr;

cout << *(ptr2) << endl;

cout << *(ptr) << endl;
```

*** Double Dereference operator**

*(Ptr2) → 100
**(Ptr2) → 10 (Double Dereference).

NULL Pointer :-

→ A pointer that doesn't point to any location.

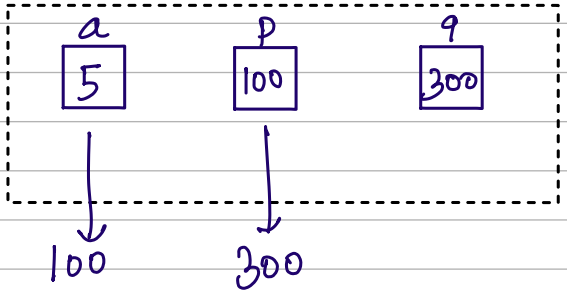
- Defining null pointer

int* ptr = NULL; → gives :- 0x0

- We can not Dereference the "null Pointer" [Segmentation error]

Practice Question:- Predict Output.

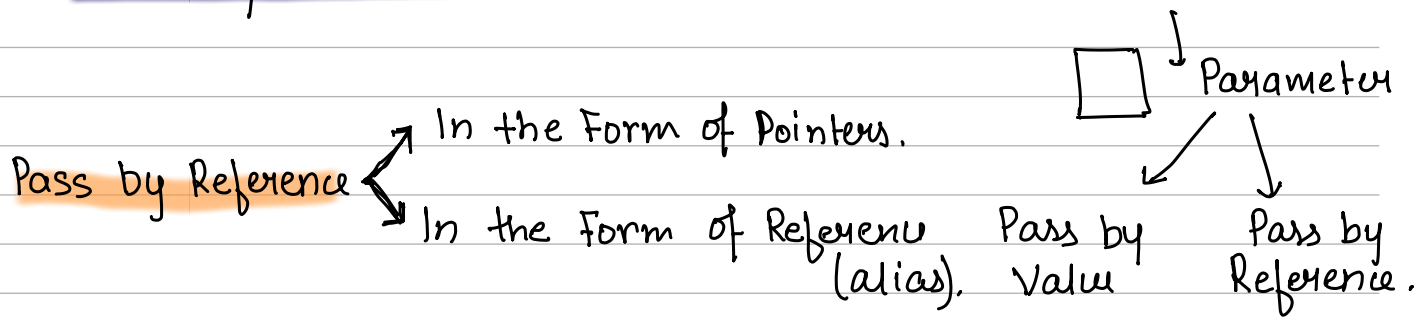
```
int a = 5.
int *p = &a;
int **q = &p;

cout << *p << endl;
cout << **q << endl;
cout << p << endl;
cout << *q << endl;
```

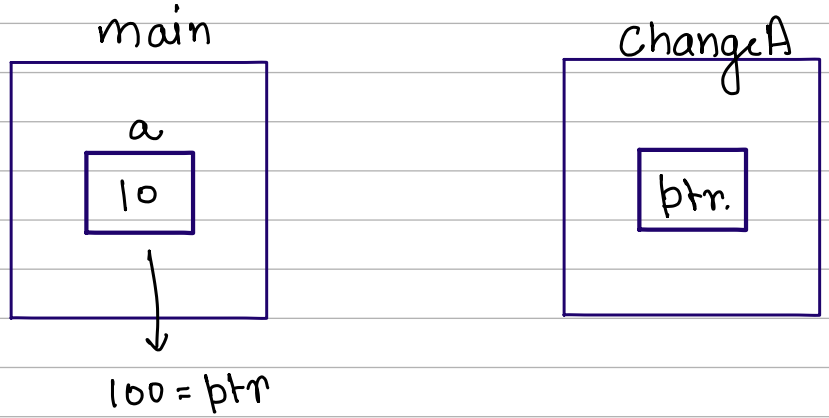


output :-

1. 5
2. 5
3. 100
4. 100

Pass by Reference.





Example:

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

void changeA(int*a) {

    *(&a)=20;
}

int main() {

    int a=10;

//changeA(a); ⟶ Pass by Value
 changeA(&a);
 cout << "inside the main Function:" <<a << endl;
```

Output:

inside the main Function : 20

• REFERENCE (alias).

alias ⇒ other name of same person

Iron-man ↓ same person ↓ Tonystark

```
void changeA(int &b){
    b=20;
}
```

"b" is alias of b.

```
int main(){
    int a = 10;
    changeA(a);
    cout << a << endl;
}
```

⟶ "20"

Array Pointer :

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

0	1	2	3	4
1	2	3	4	5

"arr" ⟶ name of arr AND A special pointer.

therefore.

```
cout << *arr << endl;
```
⟶ this gives Arr[0] = 1.

⇒ arr is also called as constant pointer.

```
int arr[] = {1,2,3,4,5};
int a = 15;
arr = &a;   // this gives error because this is constant Pointer.
```

Pointer Arithmatic

- Increment (++) / Decrement (--).

Ptr++ → it means Ptr+1

lets suppose Ptr is Pointer of Int type.

then 1 adds 4 bytes in original Ptr.

∴ Ptr++;

original Ptr = 100
after Ptr++ → 104.

→ same goes for Ptr--;

```
ptr= 100;        // Suppose
Ptr--;
cout << Ptr << endl;   → it gives 96.
```

- Add / Subtract

→ Ptr = Ptr+2

original Ptr = 100

after "Ptr+2" = 108 // if Ptr is of int type.

→ Ptr = Ptr-2

original Ptr = 100

after "Ptr-2" = 92 // if Ptr is of int type.

Note:- We can not add two pointers but we can subtract 2 Pointers.

- Subtract Ptr
  → if Ptr1 & Ptr2 is on same datatype(int) then (Ptr2 - Ptr1) shows how many integers space is there.
  ∴ let →
  Ptr 2 = 104
  Ptr 1 = 100
  then Ptr2 - Ptr1 = 1 (1 Int = 4 bytes).

- Comparision operators (<, <=, >, >=, !=, ==)

  We can Compare values of Pointers by using Comparison operators.

Binary Search Algorithm

- → Practical
- → Approach
  - ↓
  - Iterative code
  - ↓
  - optimized code
- → Time complexity of Binary Seach.
- → Recursion code For Binary Search.

• Binary Search:

→ we have already learned linear Search:

linear search.

now, Binary search is the optimised searching Algorithm as compared to linear Search Algorithm.

Time Complexity of BINARY SEARCH is

$O(\log n)$

→ linear search can be applied on any Array (sorted/unsorted).

but,

Binary search can apply only on:

- Sorted Array (Asc. / Desc.)

Example: Binary search.

arr = [-1, 0, 3, 4, 5, 9, 12] → sorted array

target = 12. ∴ n = 7.

-1	0	3	4	5	9	12

index: 0 n-1 ⟶ asc. order.

step 1: Find mid of Array:

arr[mid]

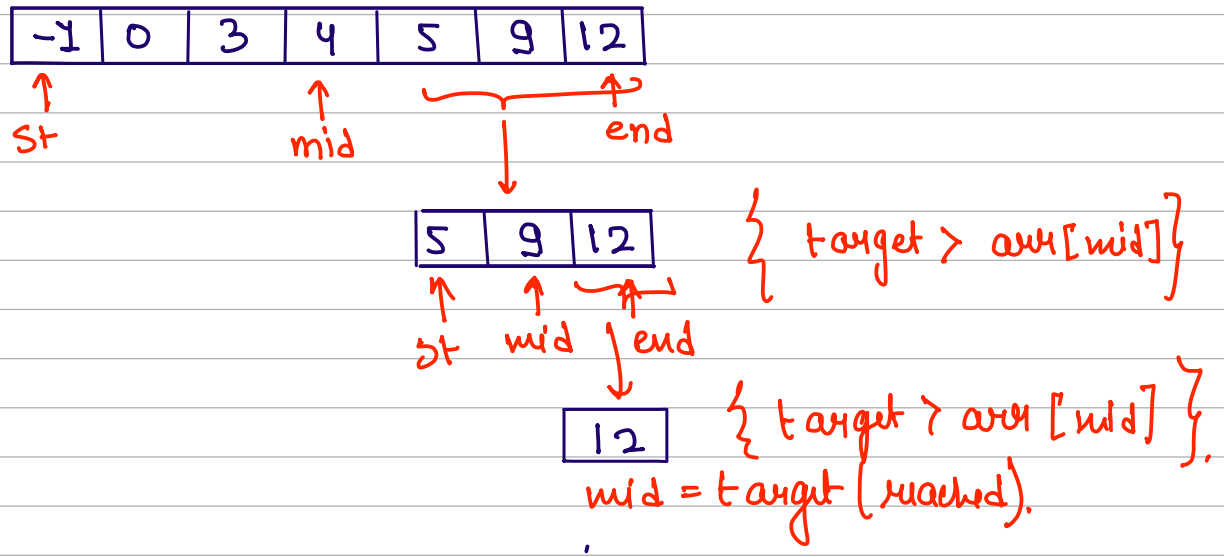
↓

$\left[\frac{st+end}{2}\right]$ =

Step 2 : target > arr[mid] → Search in → Right Side

target < arr[mid] → Search in ← left Side.

target = arr[mid] → target matched.



CODE :

```
st = 0;
end = n-1;        // n = arr.size();
→ while ( st <= end) {
    mid = ( st + end)/2.      → mid = st + (end-st)/2
  → if ( target > arr[mid]) {
        st = mid+1 ;
    }
  → else if ( target < arr[mid]) {
        end = mid-1;
    }
  → else {
        return mid ;
    }
}
return -1;
```

For the worst case cenirio.

St = Int_MAX

END = Int_MAX

So, when we add Int mid = [(st+end)/2] Int can't Store So this cause overflow

→ to Solve this we calculate mid by using Formula.

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

(•) Time Complexity of Binary Search Approach:

In leaner Search
↓
"For loop".
→ we know
if array is of size. n
then we need to perform
n iteration to find
our target value using
linear Search.
∴ let n=10
we need 10 iterations.

In Binary Search
↓
array of size n
∴ let n=10
then, no. of iteration.
n=10
↓
n=5
↓
n=2
↓
n=1
} 4 iteration

∴ For n=10 (size of array)

In Linear Search
↓
no. of iteration = 10

In Binary Search
↓
no. of iterations = 4

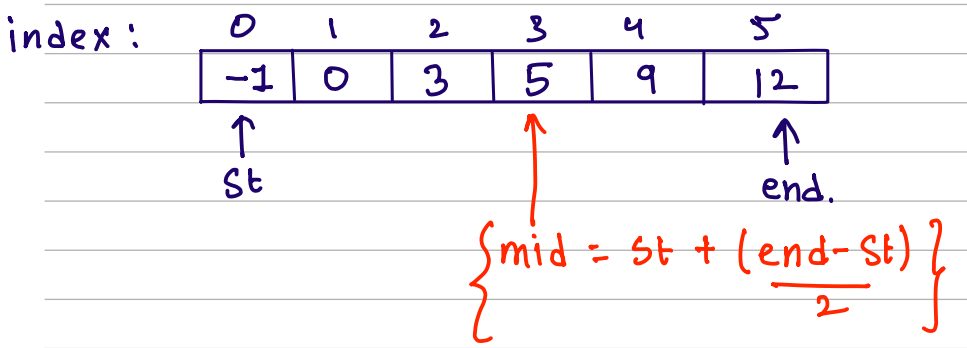
For worst case Scinio

→ Time Complexity:→

n : $n/2^0$ → $\frac{n}{2^k} = 1$
↓
$n/2$: $n/2^1$ → $n = 2^k$
↓
$n/2^2$: $n/2^2$ → $[\log_2 n = k]$
↓
⋮
1. → $n/2^k$ ↑

hence, the time complexity of binary search algorithm is $O(\log n)$

Binary Search using Recursion



- Code:

```
int recbinarySearch(vector<int> arr, int target, int st, int end){
    if( st <=end){
      int mid= st + (end-st)/2 ;
   if( target > arr[mid]) {    // 2nd half
        return recbinarySearch (arr, target,mid+1, end);
    }
   else if( target < arr [mid]) {  // 1st half
        return recbinarysearch(arr, target, st, mid-1);
    }
   else {    // mid = answer
       return mid;
    }
  }
   return -1;
}.
```

→ Binary Search Questions on leetcode and Github (KrrishBajwa).

note: Video no. 21 (Book Allocation Problem)
Video no. 22 (Painter's Partition Problem). } Pending
Video no. 23 (Aggressive Cow Problem).

Important

Sorting

Sorting :

Sorting is basically arranging our data in Particular order.

e.g 5, 3, 1, 2

↙ ↘

1, 2, 3, 5 5, 3, 2, 1

Algorithms for Sorting :

1. Bubble Sort :

Basic funda. {
- (n-1) iteration
- ↓
- adj. el compare
- ↓
- larger last me push (using swap).

• Pseudo code :

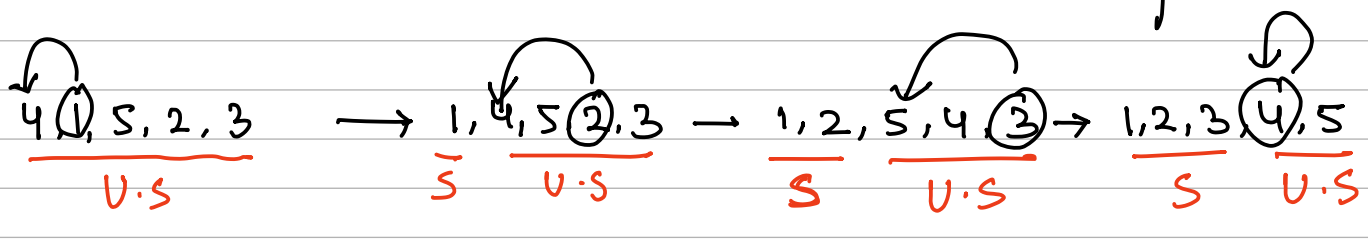
```
For(int i=0 ; i<n-1 ; i++){
    For(int j=0; j<n-i-1 ; j++){
        iF( A[j] > A[j+1]){
            swap(A[j], A[j+1])
        }
    }
}
```

* Time Complexity of Bubble Sort = $O(n^2)$

Selection Sort :→

[4,1,5,2,3]. n=5

* we need n-1 iteration to sort this array



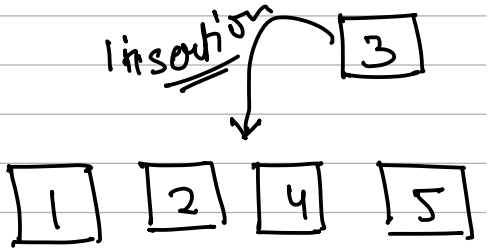
Sorted Array. → 1,2,3,4,5

Code:

```
For (i=0; i<n-1; i++){
    int SmallestIndex = i;   // to compare with others.
  for (j=i+1; j<n ; j++){
      if( arr[j]< arr[SmallestIndex]){
          SmallestIndex = j ;
      }
    }
    Swap(arr[i], arr[SmallestIndex]);
}.
```

3. Insertion Sort

[4,1,5,2,3]



Code:

```
For (i=1; i<n; i++){
int   Current = Arr[i];
int   Previous = i-1;
    while (Previous >=0 && A[Previous]> A[Current]) {
        A[Previous+1] = A[Previous];
        Previous--;
    }
    A[Previous+1] = Current;
}
```

* Next Permutation Question.

1. Pivot Element ⇒ A[i] < A[i+1] {Find Pivot Element}
2. Find the right most element > Pivot Element

   Swap (RME, Pivot).
3. Reverse all Elements after index of [Pivot+1] Element.

e.g. 1 2 3 6 5 4

Step 1

← increasing order an "3" is the First Element at any index "i" so, that

- Arr[i] < Arr[i+1]

Pivot element

Step 2

← Right most element > Pivot

→ now in this section we need to Find the element just larger than the Pivot Element

And Swap (3, 4)

Step 3

to make this number smalles we just need to reverse this Part of array. (Decreasing to Inc.).

6 5 3 ⟶ 3 5 6

1 2 4 3 5 6 → this is the final Answer

different Example.

1, 2, 5, 4, 3

1. Pivot Element

1, 2, 5, 4, 3 (2 < 5) A[i] < A[i+1] } ← From Backward }

↑ P.E ↑ R.M ele.

2. 1, 3, 5, 4, 2

↳ Reverse this Part.

1, 3, 2, 4, 5 ⟶ Ans.

Code:

```
                        // Find Pivot.
Piv = -1;
For( i = n-2; i >= 0 ; i--) {
    if ( Arr[i] < A[i+1]){
        Piv = i;
        break.
    }
}
if(Piv == -1) {
    Reverse Array;
    return;                 // in Place changes.
}
                        // 2nd largest Element.
For( int i = n-1;  i > pivot ; i--) {
    IF( A[i] > A[Pivot]) {
        swap( A[i] , A[Pivot]);
        break;
    }.
```

// 3rd step: Reverse element from (pivot+1) to n-1.

STRING

String in c++:

→ In c++, String are sequence of characters that are used to store words & text.

They are also used to store data, such as number and other types of information in the form of text. String are provided by <string> header File in the Form of std::string class

- Before learning about string we have a topic "character Arrays".
  - CHARACTE ARRAY
    - ↳ also known as c-Strings.

→ initialization

```
char Arr_name[] = {'a', 'b', 'c', 'd', 'e'};
```

```
char Arr_name = {'a', 'b', 'c', 'd', '/0'};
```

→ `char Arr_name[];`

`cin >> Arr_name ;`

if we enter "Hello" then it print "Hello".

if we enter "Hello World" then it print "Hello".

note: cin ignore character after space.

so, there in special function for string input

`cin.getline(str, len, delim?)`

- str → name of array
- len → no. of char
- delim → delimiter.

Note: In character arr we had a lot of limitations, then String of c++ comes in

String in c++

→ String Str = "Hello krish";

- String: this is just use as a normal data type
- Str: name of our string (we don't need to define the size for string)
- "Hello krish": our string

★ note: our strings in c++ are dynamic in nature.

- **String supports:**

1. String Str3 = Str1 + Str2; // concatigation.
2. Str1 == Str2; // Comparision
3. Str2 > Str1; // Compare Lexicographically
4. Str.length(); // return length of string.
5. getline(cin, Str); // to take input of string
6. "for loop" & "for-each loop" are applied similar to array

* Reverse a String

1. For character arr. Charr =

H	E	L	L	O	\0
↑ st				↑ end.	

```
st = 0;
end = charr.size() - 1;
while (st < end) {
    swap(charr[st], charr[end]);
    st++;
    end--;
}
```

* Problem no. 344 on leetcode.

2. For String

- using STL Function "reverse".

```
String  Str = "Krish Balana";

reverse (Str.begin(), Str.end());
```

→ they return iterators.

```
Cout << Str << end;
```

Q. Find if the String is palindrom or not.

```
# include <iostream>
# include < Algorithm>
using namespace Std;

int main(){
  String Str1 = "BOB";
  String Str2 = Str1;
  reverse (Str2.begin(), Str2.end());
  if( Str1 == Str2){
     cout << "Yes Palindrom";
  }else{
     cout << "not";
  }.
```

Some Important Question related to String

1. Valid Palindrome

```
S = "Ac3?e3c?a";
```

- key point
  1. uppercase & lowercase of same alphabet will be considered as same
  2. Special character are not considerable

3. Permutation in Strings { VIDEO NO 31 ON DSA SERIES OF "APNA COLLEGE" } // Problem no. 567 in leetcode

s1 = "ab"; s2 = "eidbaooo";

we need to tell either s1 or any other permutation of Stein s1 exist in Sking s2.

Approach:

1. Store the frequency of characters in s1.

```
int Freq[26] = {0};
Freq[s1[i] - 'a']++
```

↳ Gives the ASCII Value.

2. Search s1 permutation in s2, window Based approach.

→ create window Frequency array to match the window Frequencies with s1 Frequencies.

Code

```
bool checkInclusive(String s1, Sking s2) {

    int Freq[26] = {0};
    For (int i=0; i < s1.length(); i++) {
        int idx = s1[i] - 'a';   // a→0, b→1, c→2
        Freq[idx]++;
    }
    int windowSize = s1.length();

    For(int i=0; i<s2.length(); i++){
        int windowIdx = 0;
        int idx = i;
        int WindowFreq[26] = {0};
        while(windowIdx < windowSize && idx < s2.length()){
            WindowFreq[s2[idx] - 'a']++;
            WindowIdx++; idx++;
        }
```

```
if (isFreqSame (Freq, windowFreq)) {
    return true;
}
}
return false;
};
}.
```

Func" to track this

```
bool isFreqSame (int Freq1[], int Freq2[]){
for (int i=0 ; i< 26; i++){
if (Freq1[i] != Freq2[i]) {
    return false;
}
return true;
}.
```

LEACTURE NO. 32 & 33 CONTAINS 2 MORE PROBLEMS ON STRING WHICH IS PRESENT ON LEETCODE :- 1051, 443.

MATH FOR DSA

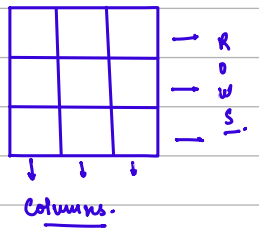
- What we'll Cover

1. Prime Logic & Sieve of Eratosthenes
2. GCD with Euclid's Algorithm & LCM
3. Digits in a number.
4. Reverse a number.
5. Check Palindrom
6. Armstrong Logic
7. Modular Algorithm
8. Fast Exponantial.

2D ARRAYS

2D-Array: rows, columns.

↳ Matrix



initialization:

```
int matrix[4][3];
```

columns. ↓ ; rows ↑

→ Initialization:

```
int matrix[Row][column];
```

e.g:

Row = 4

1	2	3
4	5	6
7	8	9
10	11	12

Column = 3.

→ `int matrix[4][3] = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}};`

— Access any element

1. In array :-

`arr[index]`

2. In 2D-array / Matrix :-

Cell ko travel karwana hota hai

`mat[r][c]`

R_0	1	2	3
R_1	4	5	6
R_2	7	8	9
R_3	10	11	12
	C_0	C_1	C_2

$R_0C_0 = 1$ $R_1C_0 = 4$

$R_0C_1 = 2$ $R_1C_1 = 5$

$R_0C_2 = 3$ $R_1C_2 = 6$ - - -

And so on.

2D-Vectors.

→ As we have 2D-Array we also have 2D-Vectors.

Jaise 2D-Array Dynamic hote hai vaise hi 2D-vector bhi dynamic hote hai.

- Initialization:

```
vector<vector<int>> mat_name = {{1,2,3}, {4,5,6,7,8}, {9,10,11}};
```

→ when we apply nested loop on this 2D-Vector to print.

```
rows = mat.size();
columns = mat[i].size();
```

→ output:

```
1 2 3
4 5 6 7 8
9 10 11
```