**STL {Standard Template Library}in CPP –**

#include<bits/stdc++.h>

// #include <iostream>

// #include<algorithm>

// #include<climits>

// #include<string>

// #include<cctype>

// #include<vector>

// #include<set>

using namespace std;

int main()

{

    // 📒 STL stands for Standard Template Library which prives an in-built functionality of already available functions, thods, techniques and libraroes.

**// 1 - Implementing STL for Binary Search**

    // int n;

    // cout << "Mention the Array size" << endl;

    // cin >> n;

    // int arr[n];

    // cout << "What are the elemenmts - " << endl;

    // for (int i = 0; i < n; i++)

    // {

    //     cin >> arr[i];

    // }

    // // Sort

    // sort(arr, arr + n);

    // cout << "THe inserted array with sort elements is - " << endl;

    // for (int i = 0; i < n; i++)

    // {

    //     cout << arr[i] << " ";

    // }

    // cout << endl;

    // int findelement;

    // cout << "which element you want to get" << endl;

    // cin >> findelement;

    // // Binary Search -

    // // Binary Search predefined functionality in STL -

    // // binary\_search(arr,arr+n,find) - for that eleemnt which we need to find.

    // if (binary\_search(arr, arr + n, findelement))

    // {

    //     cout << "Yes, Element found at " << endl;

    // }

    // else

    // {

    //     cout << "Not FOund" << endl;

    // }

    /\*

    Mention the Array size

    7

    What are the elemenmts -

    15 10 45 89 32 99 115

    THe inserted array with sort elements is -

    10 15 32 45 89 99 115

    which element you want to get

    99

    Yes, Element found at

    ----------------------

    Mention the Array size

    7

    What are the elemenmts -

    15 10 45 89 32 99 115

    THe inserted array with sort elements is -

    10 15 32 45 89 99 115

    which element you want to get

    75

    Not FOund

    \*/

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

**// 2 - using STL for unique and Non-repeating elements by using SET.**

    // Vector,Set & map are very useful Data Structures.

    /\* If in array I need to delete the repeating elemnents and want oonly unique elements which also in

    sorted manner then it 's little bit complecated and having more complecxity, so using STL and pre defined functions \*/

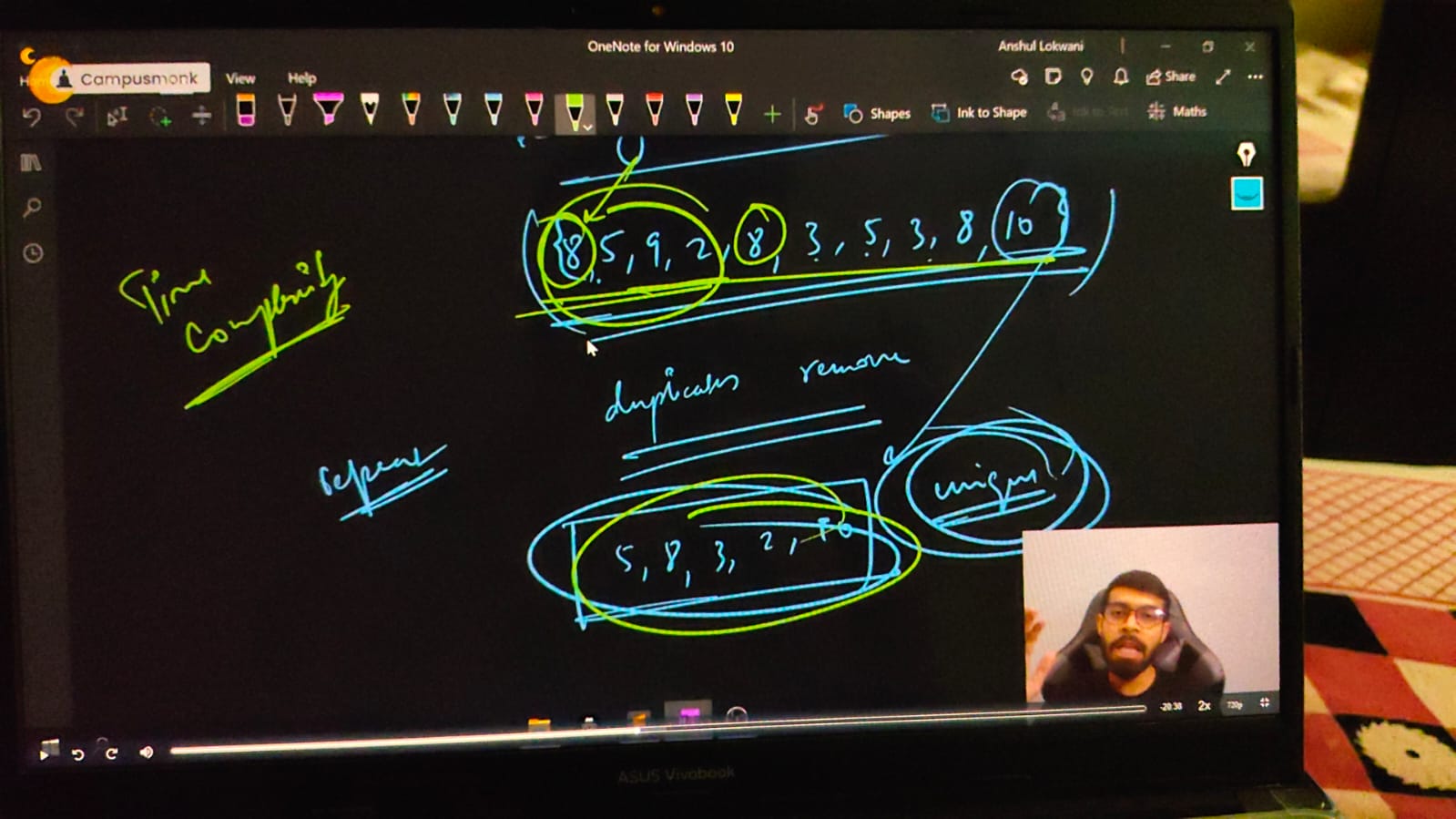
    // 📒 SET - using SET data structure we can directly store the Non-repeating and unique characters

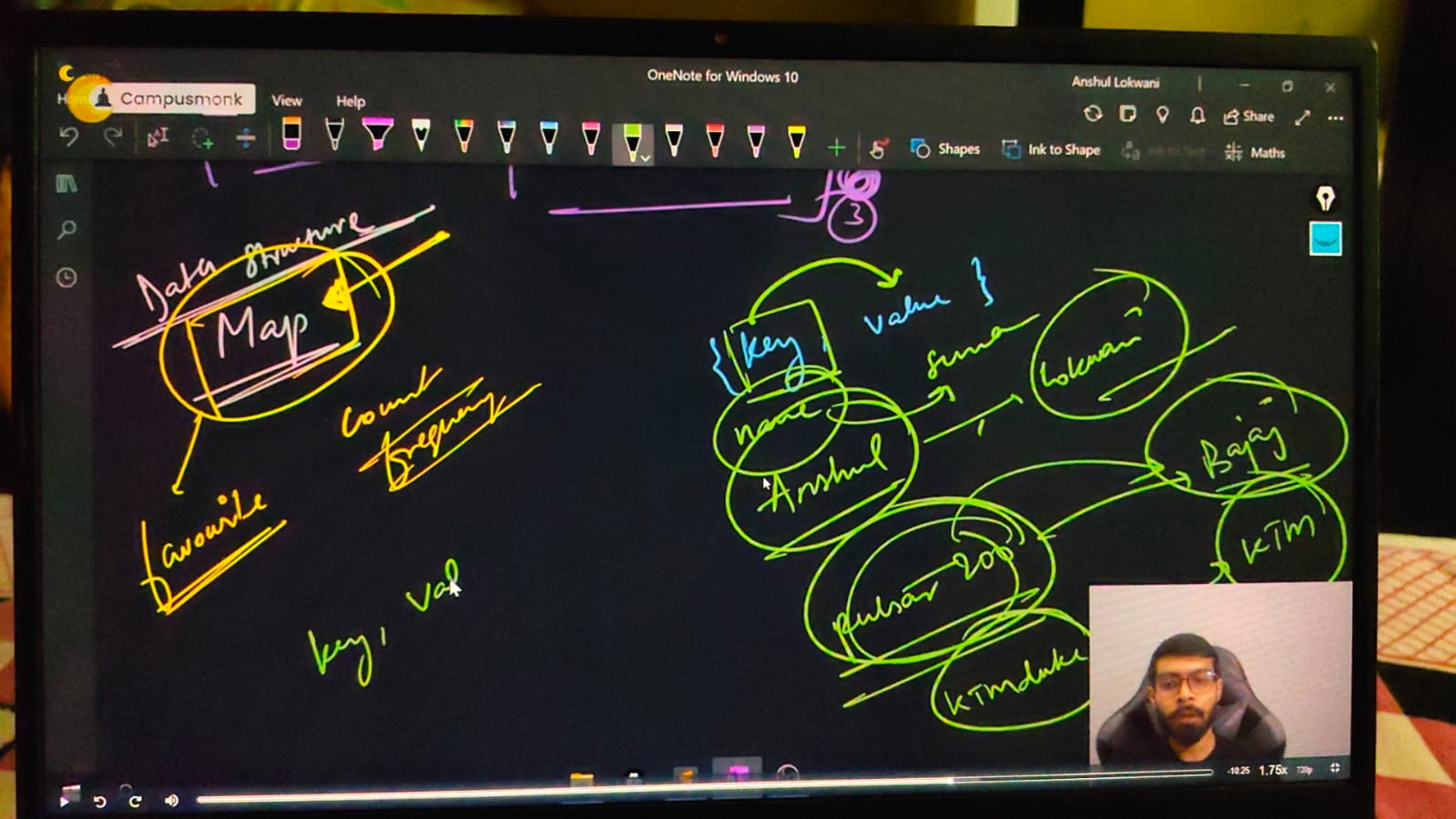
    // 📒 VECTOR - Vector is also an array but specififcally it's an DYNAMIC ARRAY. A normal array is Static Array where size can't change for any operation of any element insertion or deletion, but in vector Array it's clearly possible

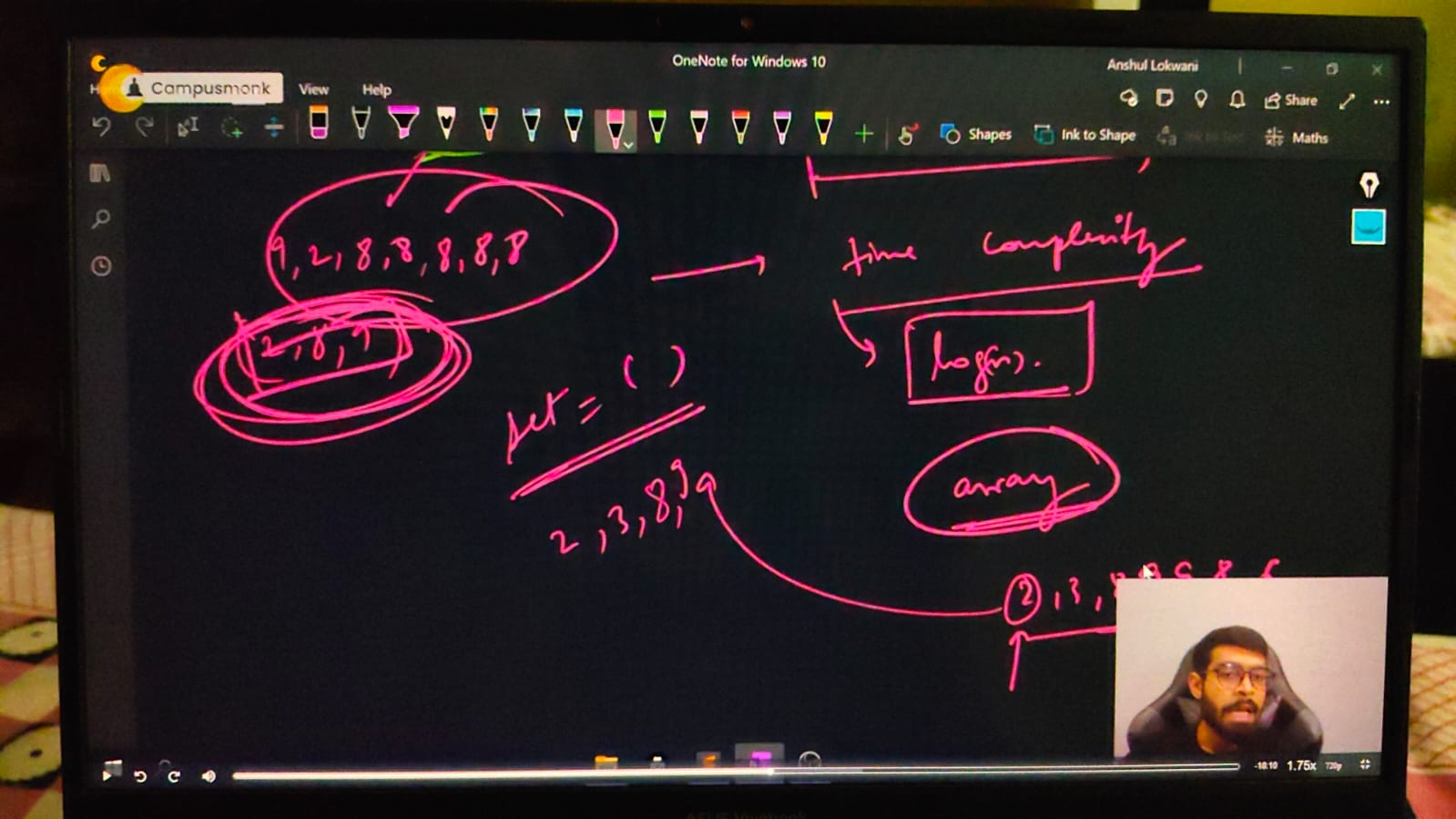
    // 📒 MAP - Easy operations like count frequesncy

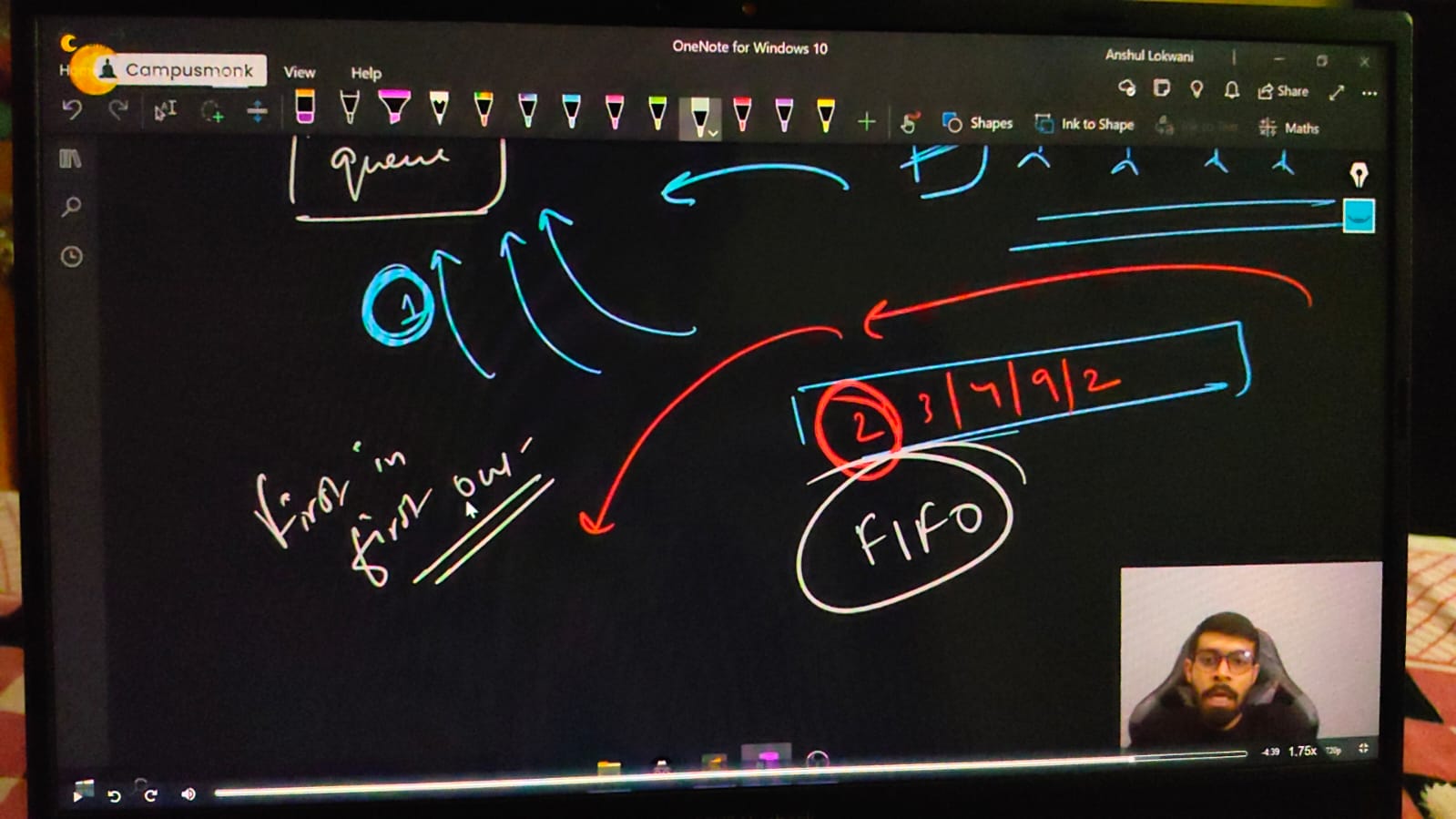
    // 📒 Queue -

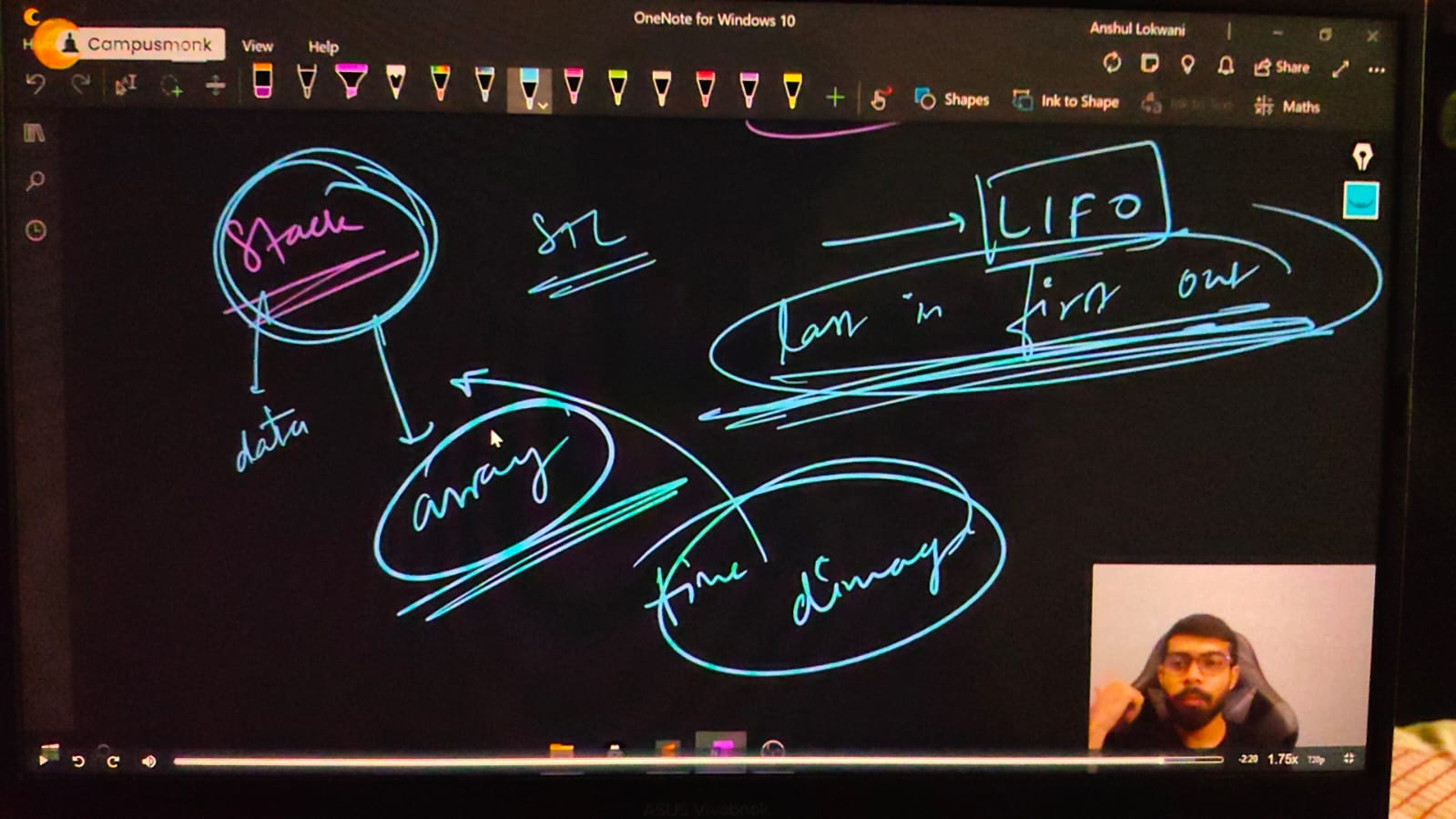
    // 📒 Stack -





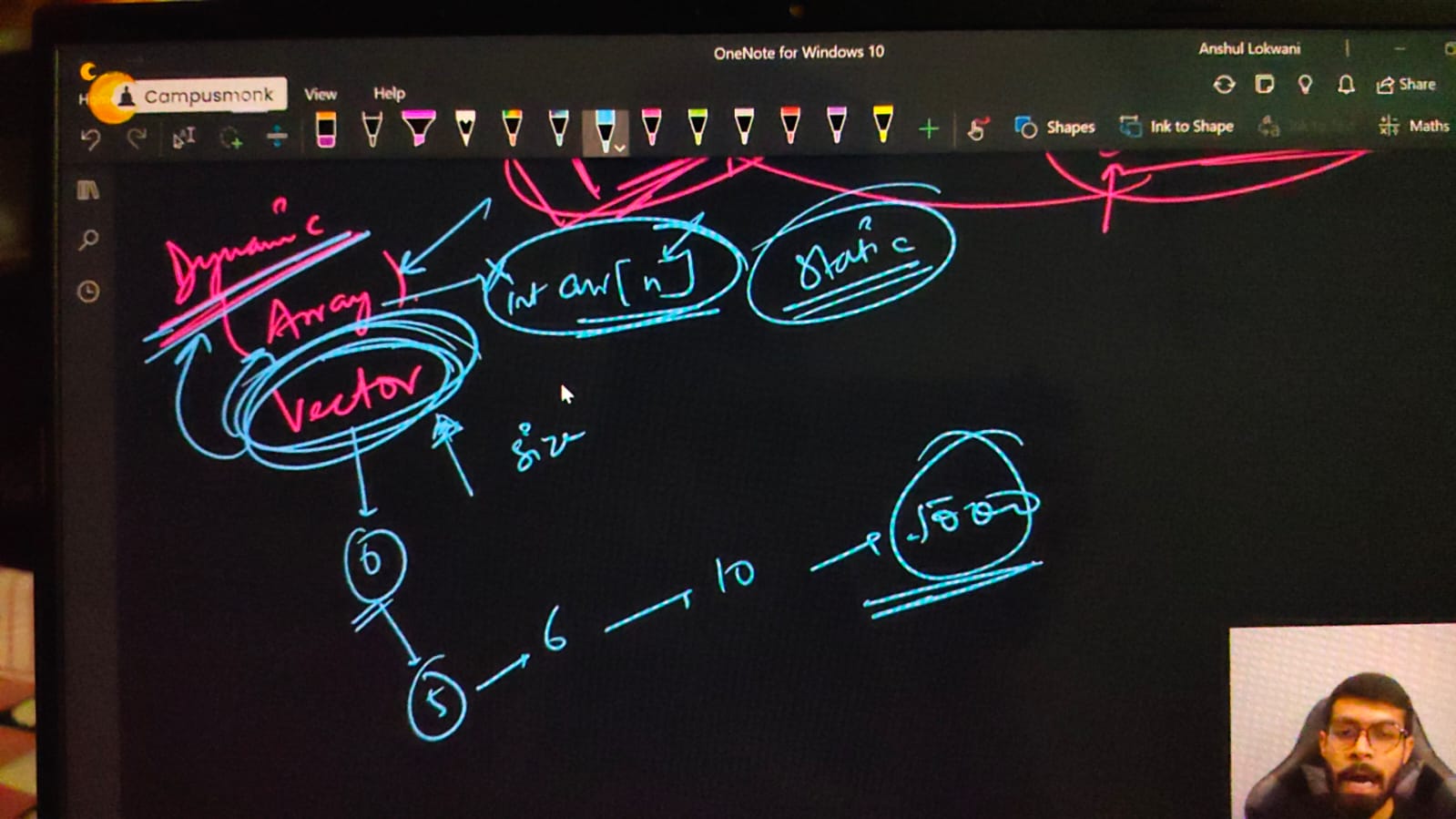


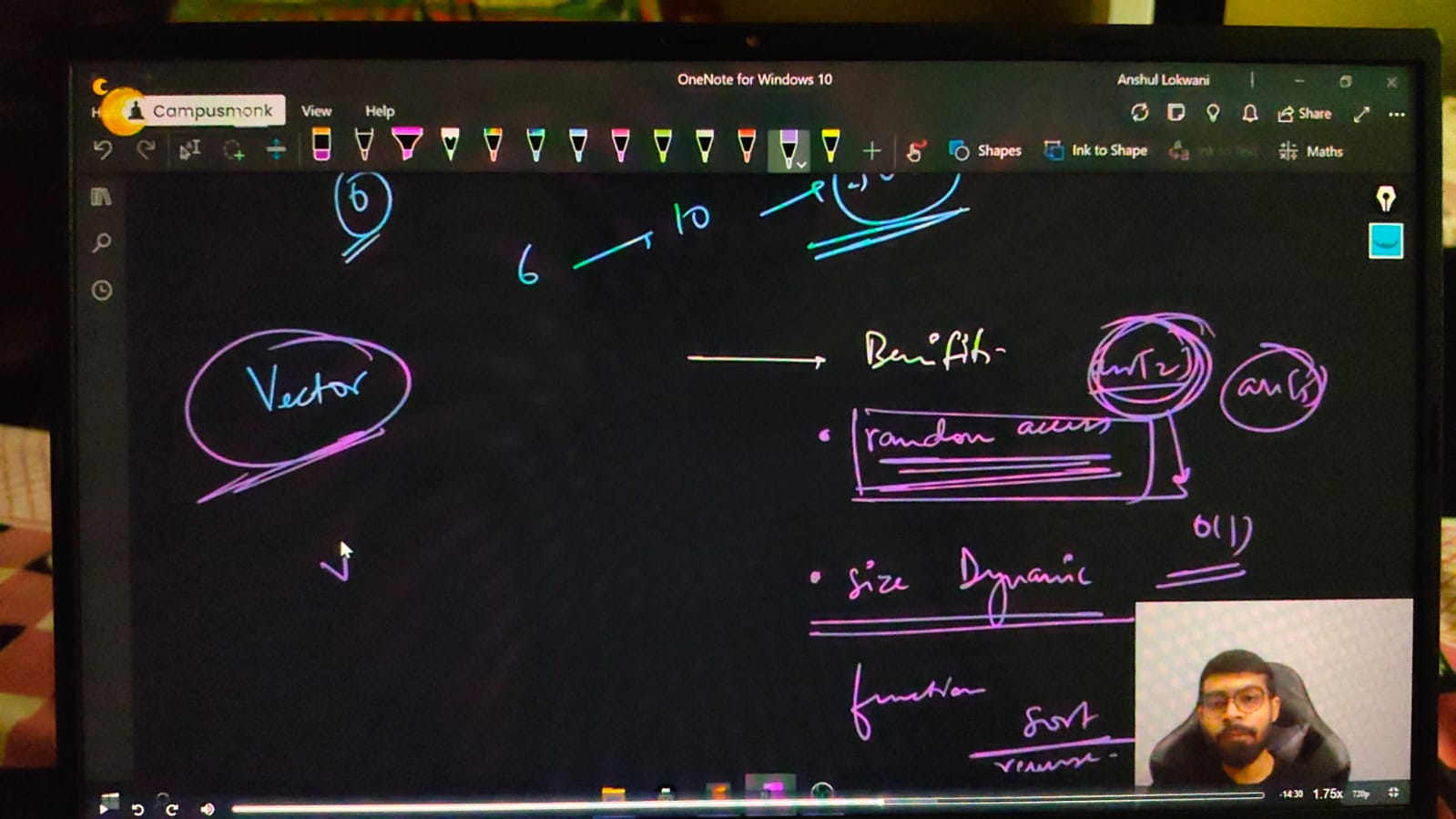


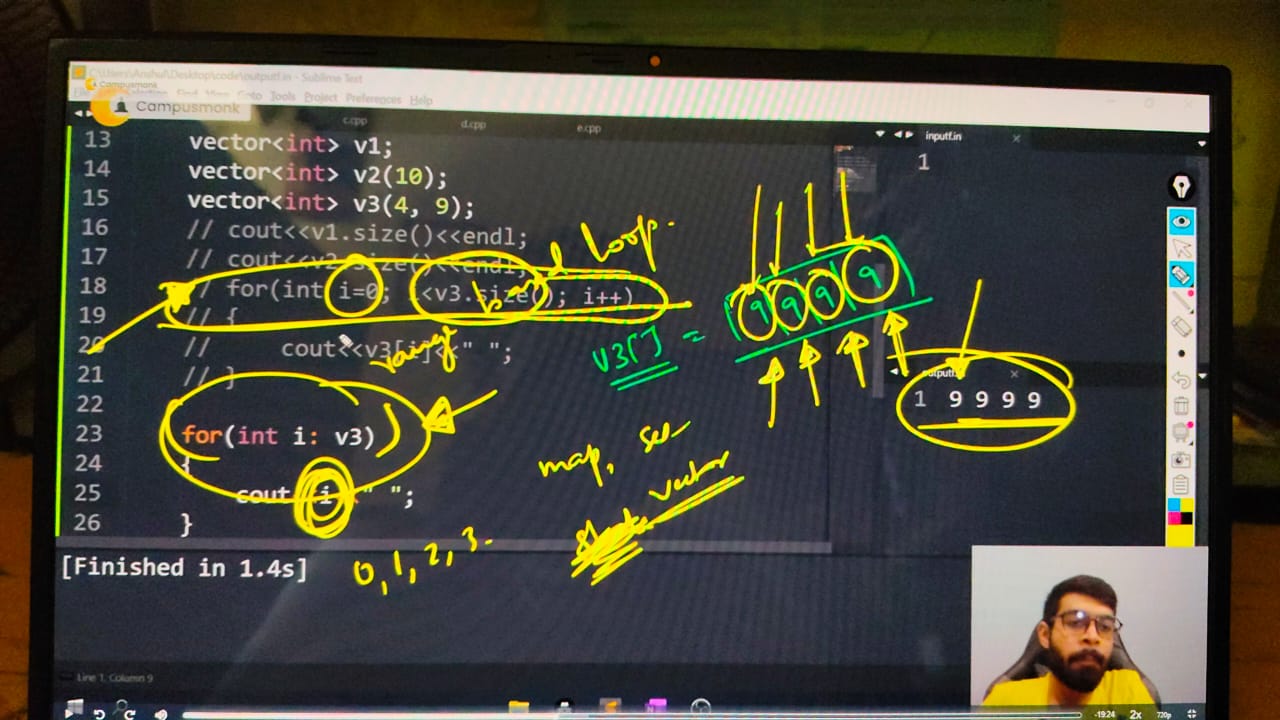


    // \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// Vector -**







    /\* **Initializing of a Vector -**

    Vector basdic syntax -

    1) vector <data\_type> vector\_name;

    2) vector <data\_type> vector\_name(size);

    3) vector <data\_type> vector\_name(size, element)

    \*/

    // vector<int> v1;

    // cout << v1.size() << endl; // 0 - by default size is always 0 in declared vector

    // vector<int> v2(5);

    // cout << v2.size() << endl; // 5 - size as per given by user for the vector

    // int arr[5]; // then it ma y include the GB - garbage value, so for avoiding this we initializze it by defaly with 0 for all elements or enter the elements manually;

    // int arr1[5] = {0};// for avoiding the garbage value we initialize by any elemnt valur or by defauly by 0

    // // Similarly in vector, after declaring the vector we use element value for avoiding the GB

    // vector<int> v3(4, 23); // all 4 elements of vector v3 having value 23

    // for (int i = 0; i < v3.size(); i++)

    // {

    //     cout << v3[i] << " ";

    // }

    // cout<<endl;

    // // 23 23 23 23

    // for (int i = 0; i < v2.size(); i++)

    // {

    //     cout<<v2[i]<< " ";

    // }

    // cout<<endl;

    // // 0 0 0 0  - agr size declare kiya h or koi elmnt ki value pass nhi ki to by default 0 for all elements

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

    // /\*by using STL here  is another way of writing for loop in short, it's known as RANGE BASED LOOPPS

    // for (data\_type i: NAMEOFVARIABLE jisme iteratekrna he)

    // {

    //     cout<<i<<" ";

    // }

    // \*/

    // vector<int> v4(4, 11);

    // for (int i : v4)

    // {

    //     cout << i << " ";

    // }

    // cout << endl;

    // // 11 11 11 11  - print all values of vector v4 using RANGE BASED FOR LOOP by STL

    // // here in this for loop i is not about 0,1,2,3...till limit. here i is clearly no. of times elements of given vector

    // vector<int> v5(4);

    // v5[0] = 20;

    // v5[1] = 81;

    // v5[2] = 111;

    // v5[3] = 15;

    // for (int i : v5)

    // {

    //     cout << i << " ";

    // }

    // cout << endl;

    // // 20 81 111 15  - printing elements i wise

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

**// Taking input in vector -**

    //     int n;

    //     cout << "The sixe of vectopr is - " << endl;

    //     cin >> n;

    //     vector<int> v6(n);

    //     cout << "Enter elements of the vector" << endl;

    //     for (int i = 0; i < v6.size(); i++)

    //     {

    //         cin >> v6[i];

    //     }

    //     cout << "So, enetered vector we have os - " << endl;

    //     for (auto i : v6)// if we're not sure about datatype then use auto for any conditon

    //     {

    //         cout << i << " ";

    //     }

    // /\*

    // The sixe of vectopr is -

    // 5

    // Enter elements of the vector

    // 23 56 89 45 12

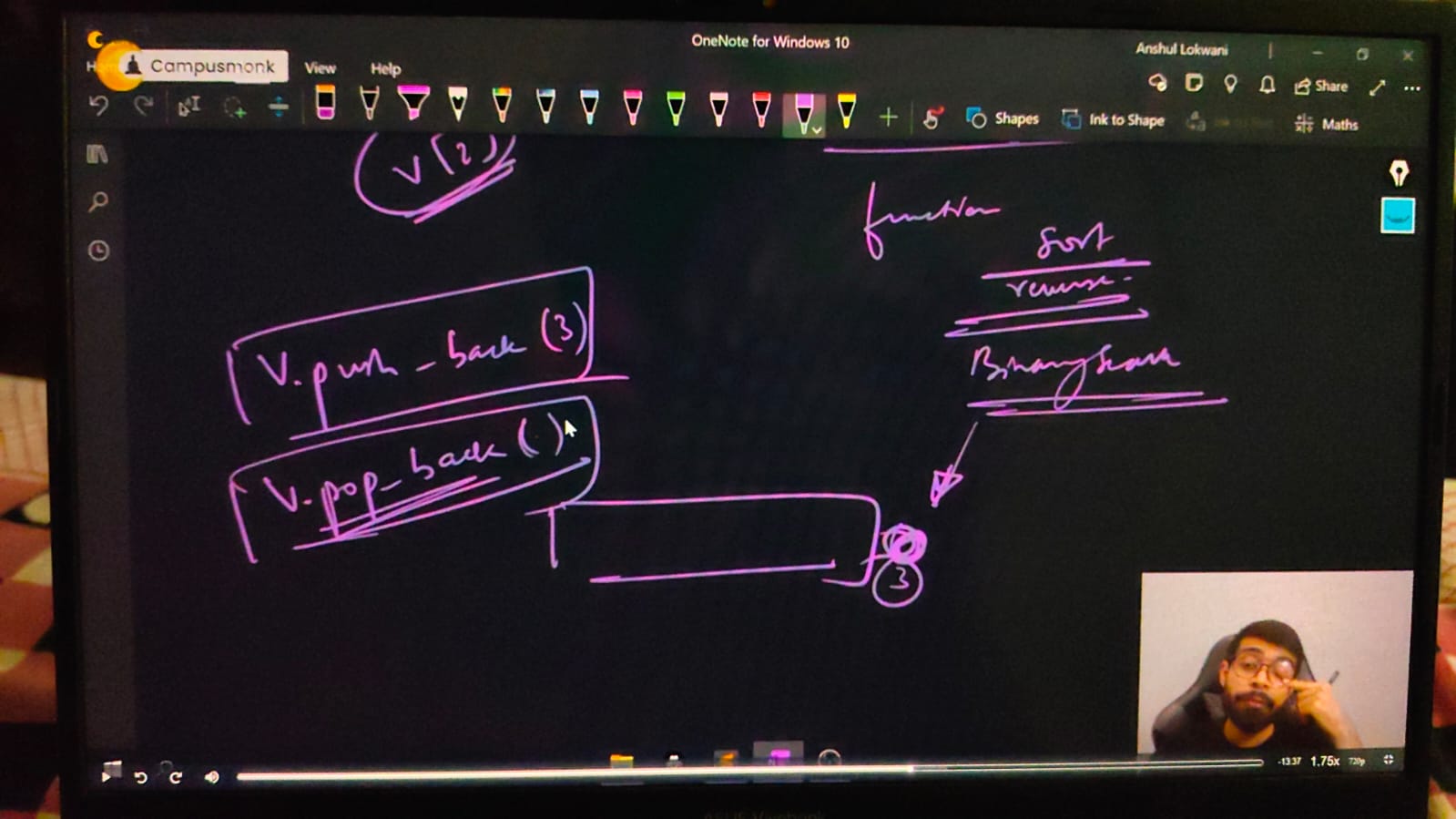
    // So, enetered vector we have os -

    // 23 56 89 45 12

    // \*/

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

**// insertion and remove in vector –**



    // vector<int> v(3, 1);

    // for (int i : v)

    // {

    //     cout << i << " ";

    // }

    // cout << endl;

    // // 1 1 1

    // // pushing elements

    // v.push\_back(3);

    // for (int i : v)

    // {

    //     cout << i << " ";

    // }

    // cout<<endl;

    // // 1 1 1 3

    // v.push\_back(5);

    // v.push\_back(10);

    // for (int i : v)

    // {

    //     cout << i << " ";

    // }

    // cout<<endl;

    // // 1 1 1 3 5 10

    // // Removing elements -

    // v.pop\_back();

    // for(int i: v)

    // {

    //     cout<<i<<" ";

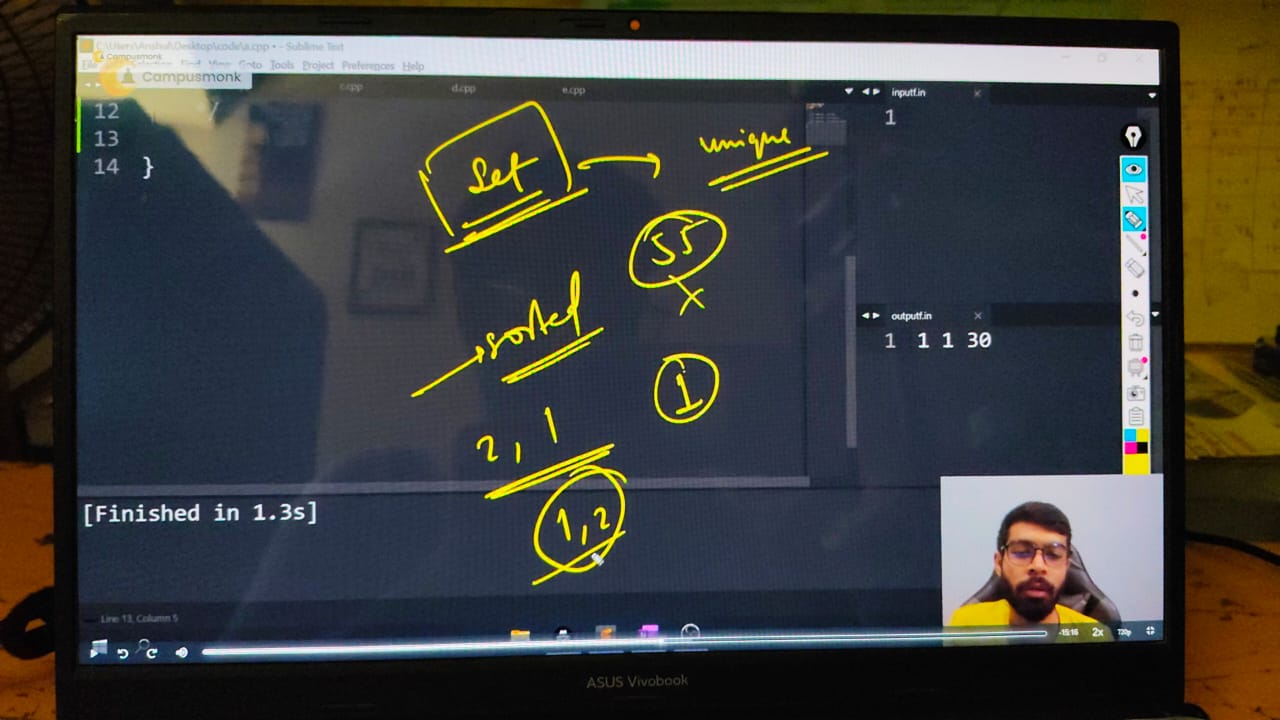
    // }

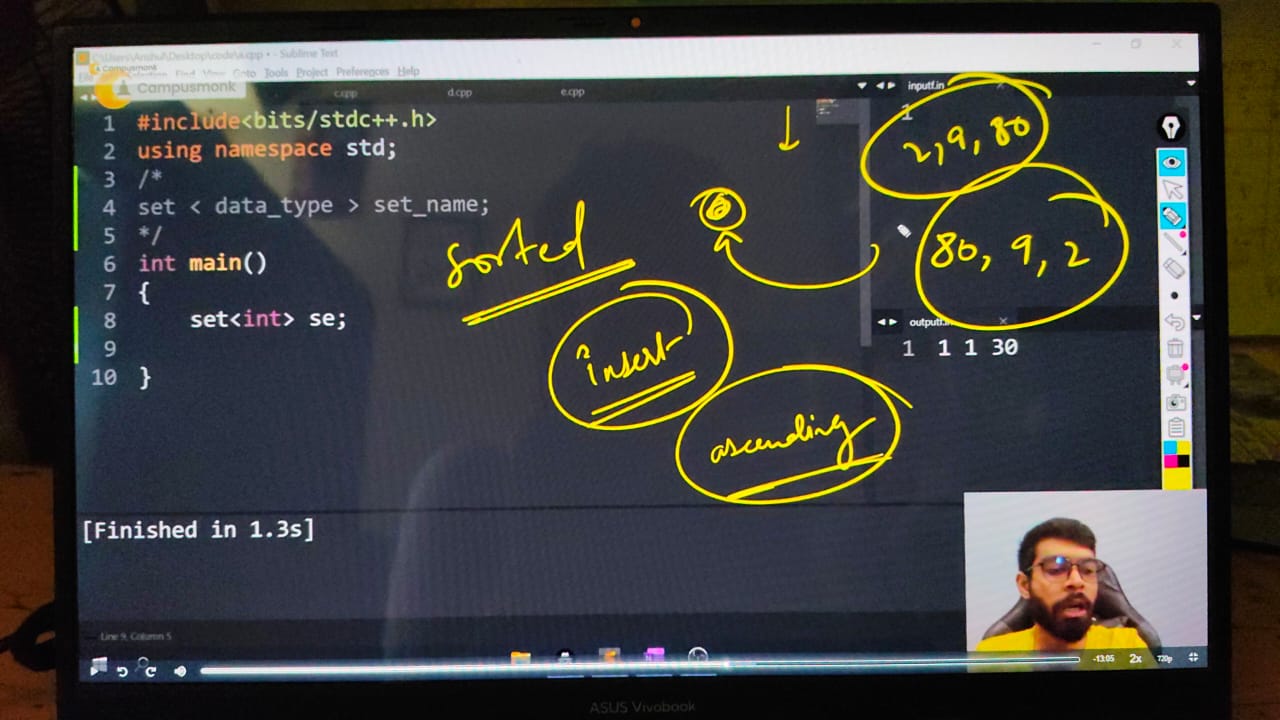
    // cout<<endl;

    // // 1 1 1 3 5  - while removing element ysing no need to mention which eloement need to remove becuase pop by defauolt removes the last elemnt only

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

**// Set –**





    /\*

    📒 - set is uses for set the elements in ascending order by default.

    In set addig eleemnt at any locaition not exist because set already elements ko set kr deta h.

    In set removing duplicate elemt is also not exist qki set by default unique eleemnts ko adscending order me arfrange kr deta h

    In set we can't use simple for loop - it doen;t exust for set function. need to use only RANGE BASED LOOPS for printing elements

    Initialization of set -

    Set syntax -

    set <data\_type> set\_name;

    inserting element in set -

    set\_name.insert(value);

    get size of set -

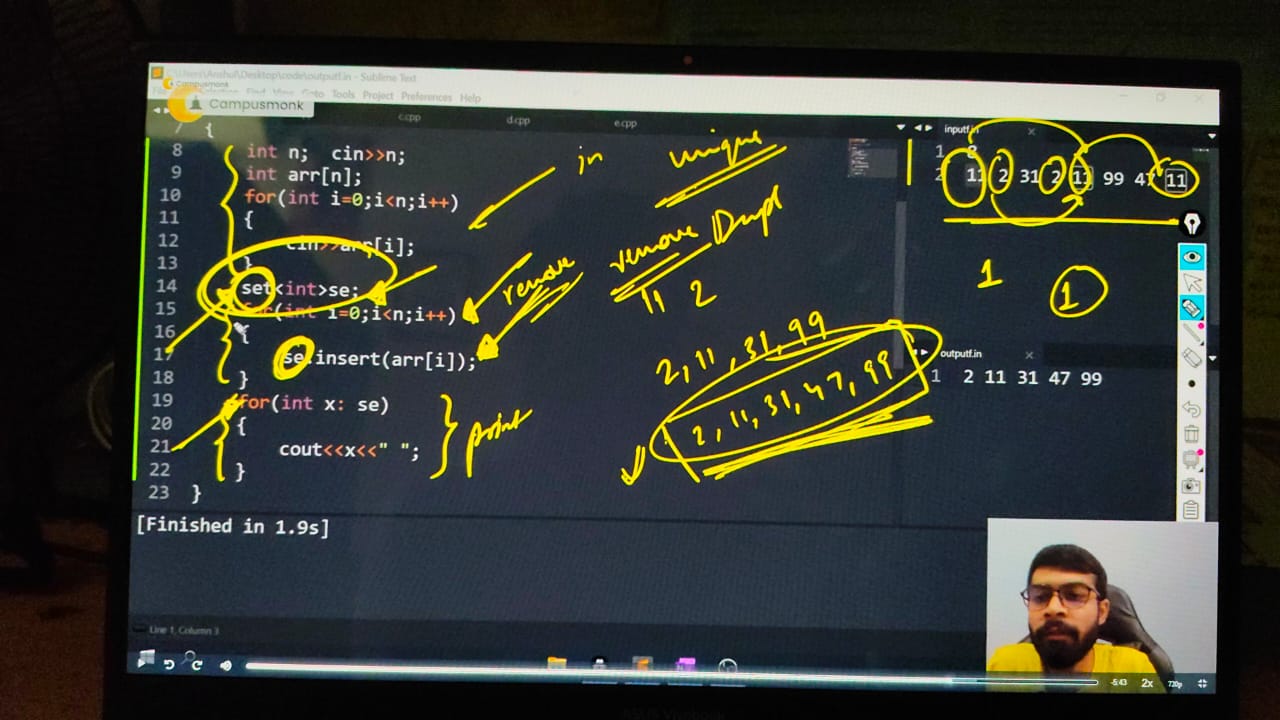
    cout<<set\_name.size();

    removing element fromset -

    set\_name.erase(valueofelementwhichneedtoremove);

    \*/

**// insert eleemnts through set**



    set<int> st;

    st.insert(1010);

    st.insert(1167);

    st.insert(24);

    st.insert(1010);

    // for printoing elemts inset -

    for (int i : st)

    {

        cout << i << " ";

    }

    cout << endl;

    // 24 1010 1167  - elemets arranged in ascedinging order by itself using set function & even duplicated element removed

**// size of set -**

    cout << st.size();

    cout << endl;

    // 3 - with unique elemets

**// remove element**

    st.erase(24);

    for (int i : st)

    {

        cout << i << " ";

    }

    cout << endl;

    //1010 1167

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

**//Qun 1) SET - Remove duplicate elements of array and arrange array in increasing order**

    // int n;

    // cout << "hey smita, Array size" << endl;

    // cin >> n;

    // int arr[n];

    // cout << "Bro, elements is also necessary to perform" << endl;

    // for (int i = 0; i < n; i++)

    // {

    //     cin >> arr[i];

    // }

    // // now for sorting w/o using sort and for duplcate w/o uisng any logic direct;ly by using SET

    // set<int> se;

    // for (int i = 0; i < n; i++)

    // {

    //     se.insert(arr[i]); // inserting all array elements into sset function

    // }

    // cout << "So, the unique elements in increasing order of array are following - " << endl;

    // for (int i : se)

    // {

    //     cout << i << " ";

    // }

    // cout << endl;

    // /\*

    // hey smita, Array size

    // 8

    // Bro, elements is also necessary to perform

    // 11 2 31 5 11 99 47 11

    // So, the unique elements in increasing order of array are following -

    // 2 5 11 31 47 99

    // \*/

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

    // // Par agar me saare elements chahta hu sorted way me but duuplicates remopve nhi krna chahta, so i'll use multiset.

    // // with duplicates -

    // multiset<int> sm;

    // for (int i = 0; i < n; i++)

    // {

    //     sm.insert(arr[i]);

    // }

    // cout << "Now, another operaiton : Printing in increasing order only w/o removing the duplicate ones" << endl;

    // for (auto i : sm)

    // {

    //     cout << i << " ";

    // }

    // cout << endl;

    // /\*

    // hey smita, Array size

    // 8

    // Bro, elements is also necessary to perform

    // 11 2 31 5 11 99 47 11

    // So, the unique elements in increasing order of array are following -

    // 2 5 11 31 47 99

    // Now, another operaiton : Printing in increasing order only w/o removing the duplicate ones

    // 2 5 11 11 11 31 47 99

    // \*/

    // // for the size of set.

    // cout << se.size(); // 6 -  return the size of set -which is all unique elements only

    // cout << endl;

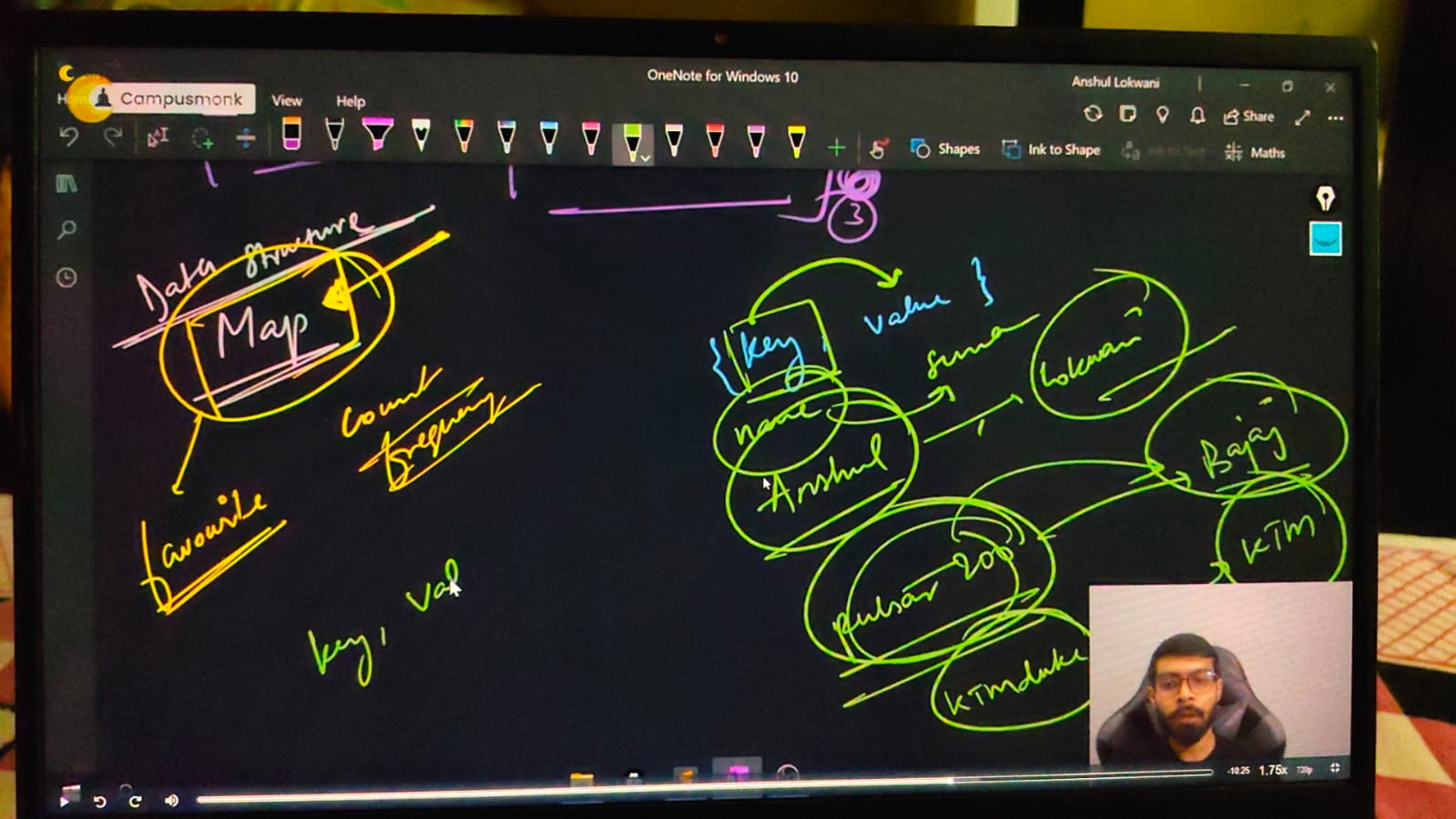
    // cout << sm.size(); // 8 - returns the size of multiset - which is of all the elemts not removed anyone includes duplicates too

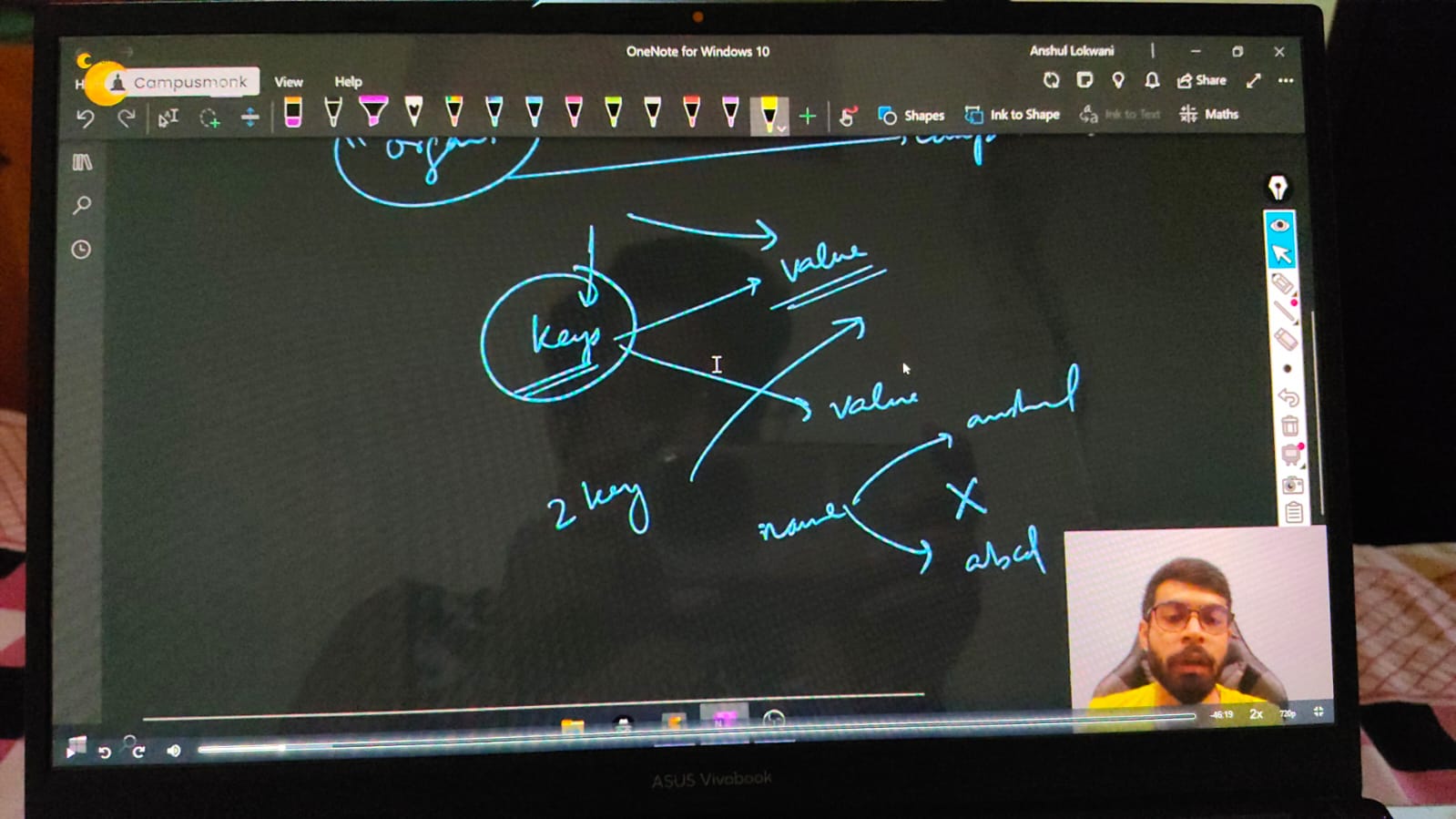
    // cout << endl;

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

}

**// Map Data Structure in CPP –**





    /\*

    📒Map is very useful data structures which takes bothh values. Key & valu Pairs. Map also sort the data in Ascending order by default

    Syntax for map -

    map <key\_datatype, value\_datatype> map\_name;

    map<key, value> mp;

    Printing values

    cout<<map\_name[key\_number]<<endl;

**map size -**

    cout<<map\_name.size()<<endl;

**remove element from map-**

    using erase function

    map\_name.erase(element)

**insert value in map -**

    map\_name.insert(make\_pair('key', Value))

OR  map\_name.insert({'Key', valupair});

    \*/

    //     map<int, string> mp;

    //     mp[51] = "Shubham Mahajan";

    //     mp[40] = "Manas Mahajan";

    //     mp[24] = "Shell Mahajan";

    // // print vallue

    //     cout << mp[24] << endl; // Shell Mahajan

    //     cout << mp[51] << endl; // Shubham Mahajan

    //     cout << mp.size() << endl; // 3

    //     cout << mp[20] << endl;    // without the value map will create a key with emepty value and added witht the mpa funtion

    //     cout << mp.size() << endl; // 4

    //     // remove value

    //     mp.erase(40);

    //     for (auto i : mp)

    //     {

    //         cout << i.first << " " << i.second;

    //     }

    //     cout<<endl;

    //     /\*

    //     Shell Mahajan

    //     Shubham Mahajan

    //     \*/

    //     // Inseret Value -

    //     mp.insert(make\_pair(22, "Ronak Mahajan"));

    //     for (auto i : mp)

    //     {

    //         cout << i.first << "-> " << i.second;

    //     }

    //     cout<<endl;

**//     // map with char datatype-**

    //     map<int, string> mp1;

    //     mp1[0] = "zero";

    //     mp1[2] = "two";

    //     mp1.insert(make\_pair(1, "one"));  // correct

    //     // or mp.emplace(3, "three");

    //     for (auto x : mp1) {

    //         cout << x.first << " -> " << x.second << endl;

    //     }

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

    //     map<int, int> mp2;

    //     mp2[1] = 0;

    //     mp2[10] = 1;

    //     mp2[100] = 2;

    //     mp2[1000] = 3;

**//     // map printing all values through simple for loop**

    //     for (int i = 0; i < mp2.size(); i++)

    //     {

    //         cout << i << " ";

    //     }

    //     cout << endl;

    //     // 0 1 2 3

    //     **// map printing all values through RANGE BASED FOR LOOP**

    //     for (auto x : mp2)

    //     {

    //         // cout<<x<<" ";

    //         // by using x directly will not able to print because x is alone here and in the map data structure its a paior

    //         // so for printing keys or values seperately we need to use dot operator for printing with allocated variable

    //         cout << x.first << " "; // 1 10 100 1000 - frits for printing the keys

    //     }

    //     cout << endl;

    //     for (auto x : mp2)

    //     {

    //         cout<<x.second<<" ";// 0 1 2 3  - for printing the values

    //     }

    //     cout<<endl;

    //     for(auto x: mp2)

    //     {

    //         cout<<x.first<<" "<<x.second<<" ";

    //     }

    //     cout<<endl; // 1 0 10 1 100 2 1000 3  - for printing keys & value pairs both

    //     for(pair<int,int> x: mp2)

    //     {

    //         cout<<x.first<<" "<< x.second<<" ";//1 0 10 1 100 2 1000 3

    //     }

    //     cout<<endl;

    //     for(pair<int,string> x: mp)

    //     {

    //         cout<<x.first<<" "<< x.second<<" ";//20  24 Shell Mahajan 40 Manas Mahajan 51 Shubham Mahajan

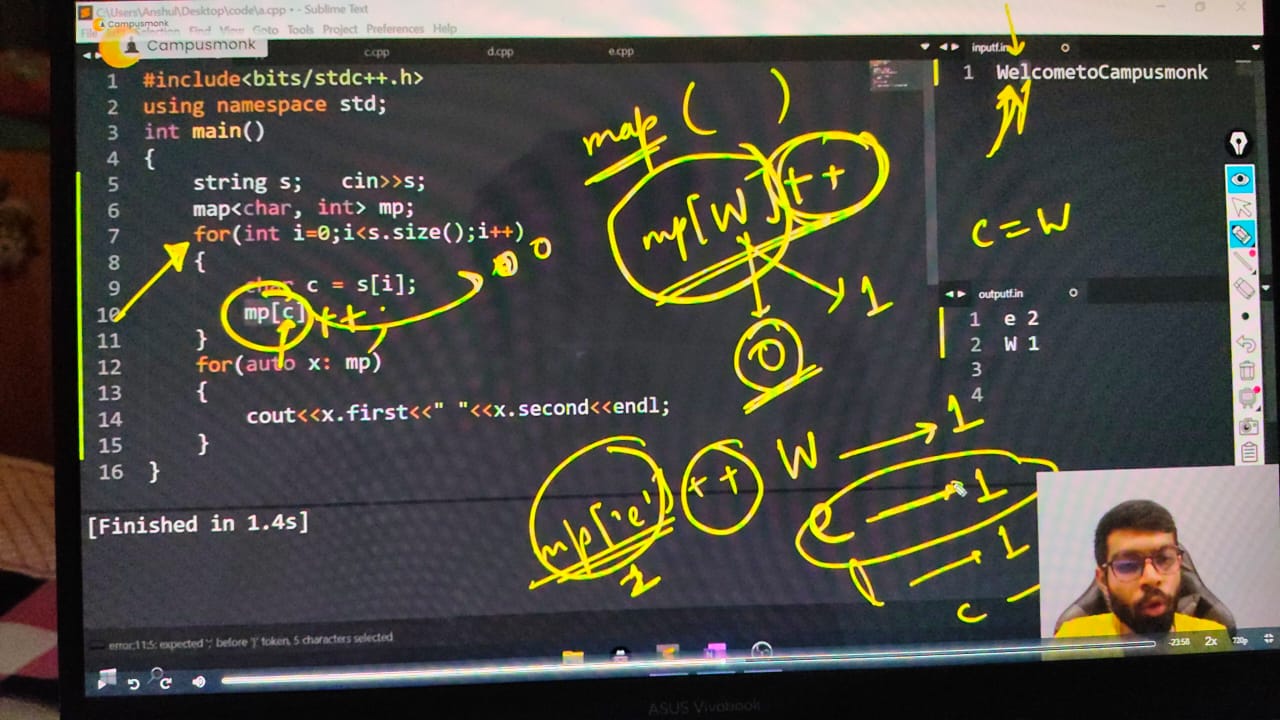
    //     }

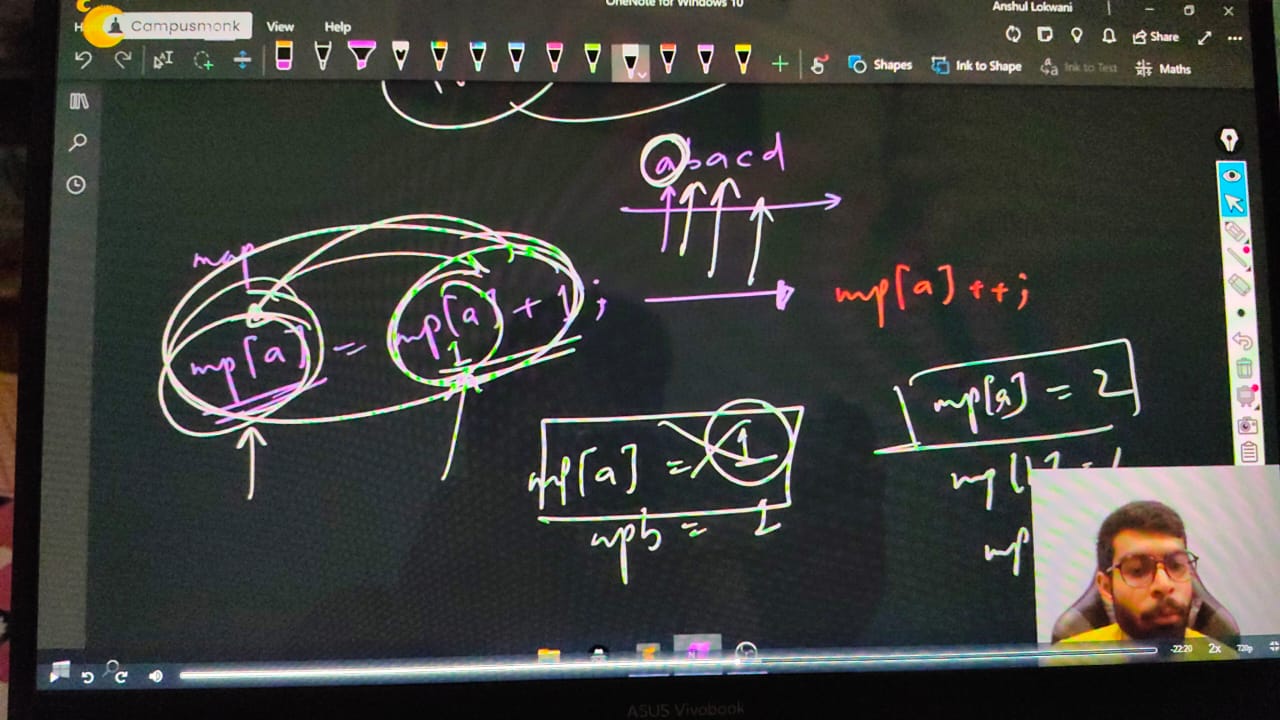
    //     cout<<endl;

    // // 📌 - so instrad of mentioning this much syntax as pair<daatatype1,datatype2> just need to use auto keyword

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

**//Qun 1) - for a string print all its characterrs that how mny times its rrpeating.**





    // string s;

    // cout << "what do you think baby..!!" << endl;

    // cin >> s;

    // map<char, int> mp;

    // cout << "the, stirng which you've inserted is - " << endl;

    // for (int i = 0; i < s.size(); i++)

    // {

    //     mp[s[i]]++;

    // }

    // cout << endl;

    // for(auto i: mp)

    // {

    //     cout<<i.first<<" "<<i.second<<" "<<endl;

    // }

    /\*

    what do you think baby..!!

    naman

    the, stirng which you've inserted is -

    a 2

    m 1

    n 2

    what do you think baby..!!

    Microsoft Hyderabad

    the, stirng which you've inserted is -

    M 1

    c 1

    f 1

    i 1

    o 2

    r 1

    s 1

    t 1

    \*/

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

**// Time complexities for VECTOR, SET & MAP -**

    /\*

    1) SET -T.C - for both Insertion & Deletion : - O(log(n)) where n - is size of SET

    2) Vector - T.C - for both Insertion & Deletion : -  constant - O(1)

    3) Map - T.c - for both Insertion & Deletion : - O(log(n)) where n - is size of SET

    \*/

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

**// Unordered Map -**

    /\*

    In ordered map chaahe =kisi bhi order me elemnts do vo hmesha increasing or ac=scending order me hi mileneg

    Leking Unorderd map me koi suroty nhi he ascending ya sorted way me aane ki

    T.C - for both Insertion & Loop up : -  constant - O(1)

    In inserttion - always the complecity is not O(1)-- iy's unorederd so elemnt found at once or sometime may be in n-times

    \*/

**// Understanding for Unoirderdd Map -**

//     map<char, int> mp;

//     mp['s'] = 26;

//     mp['e'] = 38;

//     mp['x'] = 143;

//     mp['a'] = 69;

//     mp.insert({'p', 75});

//     for (auto i : mp)

//     {

//         cout << i.first << " " << i.second << endl;

//     }

//     cout << endl;

//     /\*

//     a 69

//     e 38

//     p 75

//     s 26

//     x 143 - so using order3ed map it got print oin a order automatically , but when we use Unordfered Map -

//     \*/

//     unordered\_map<char, int> mp1;

//     mp1['s'] = 26;

//     mp1['e'] = 38;

//     mp1['x'] = 143;

//     mp1['a'] = 69;

//     mp1.insert({'p', 75});

//     for (auto i : mp1)

//     {

//         cout << i.first << " " << i.second << endl;

//     }

//     /\*

//     s 26

//     x 143

//     p 75

//     e 38

//     a 69 - Order is not fixed

//     \*/

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

//     // Look up for a particular char -

//     string s;

//     cout << "What's the next plan dude" << endl;

//     cin >> s;

//     unordered\_map<char, int> mp2;

//     for (size\_t i = 0; i < s.size(); i++)

//     {

//         mp2[s[i]]++;

//     }

//     cout<<mp2.count('s')<<endl;

//     if (mp2.count('c') > 0)

//     {

//         cout <<"c is present here";

//     }

//     else{

//         cout<<"not present buddy";

//     }

// /\*

// What's the next plan dude

// chintamani

// c is present here

// What's the next plan dude

// janhaviguta

// not present buddy

// \*/

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

// from now onwards instead of array we'll be creating always Vector -

// int n; cout<<"Enter size of vector"<<endl; cin>>n;

// vector<int> v(n);

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

// {

//     cin>>v[i];

// }

// cout<<"So, the inserted elements of vector are following - "<<endl;

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

// {

//     cout<<v[i]<<" ";

// }

// cout<<endl;

// cout<<v.size();

**Some Quns –**

**QUn -1)**

