**Sorting Algorithms –**

#include <bits/stdc++.h>

// #include <iostream>

// #include<algorithm>

// #include<climits>

// #include<string>

// #include<cctype>

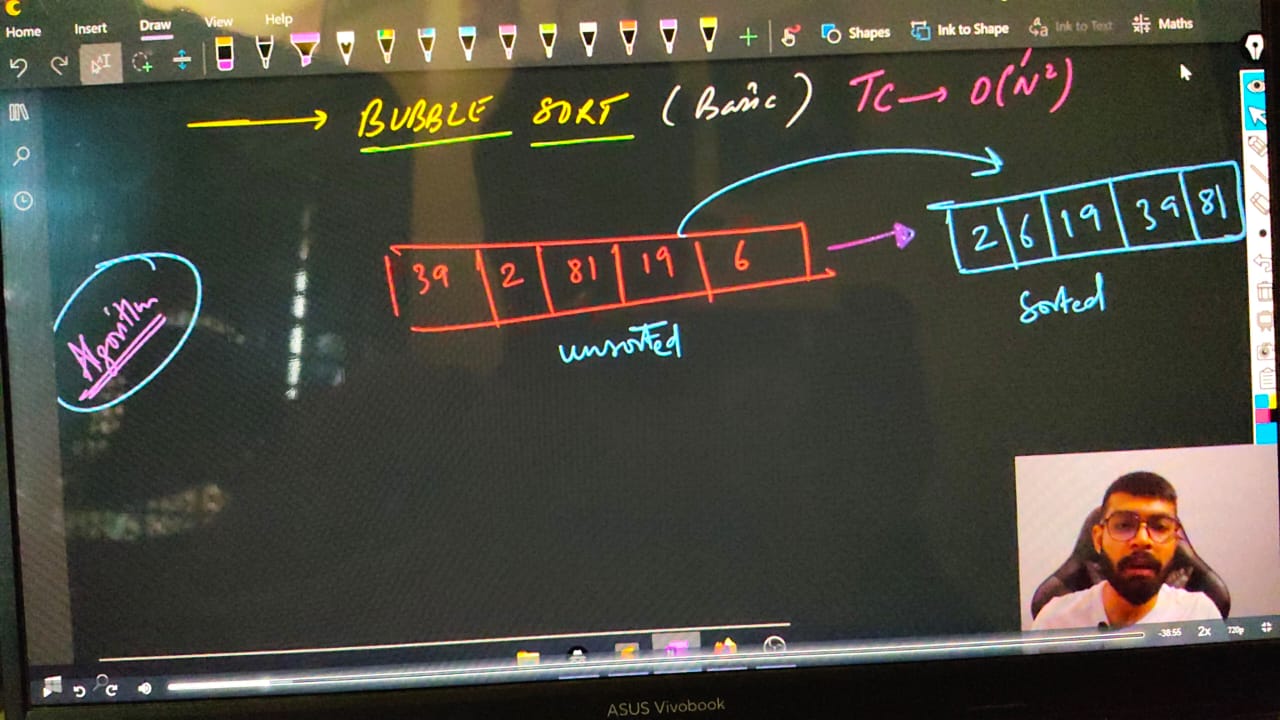
// #include<vector>

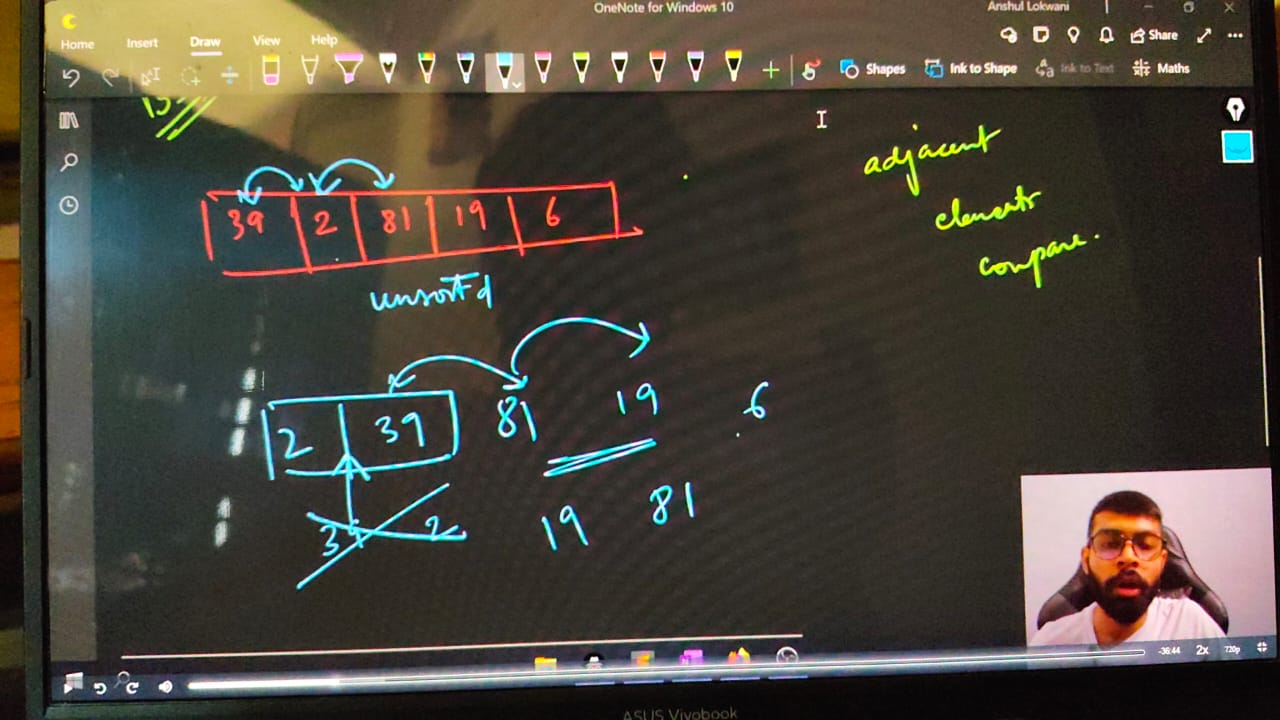
// #include<set>

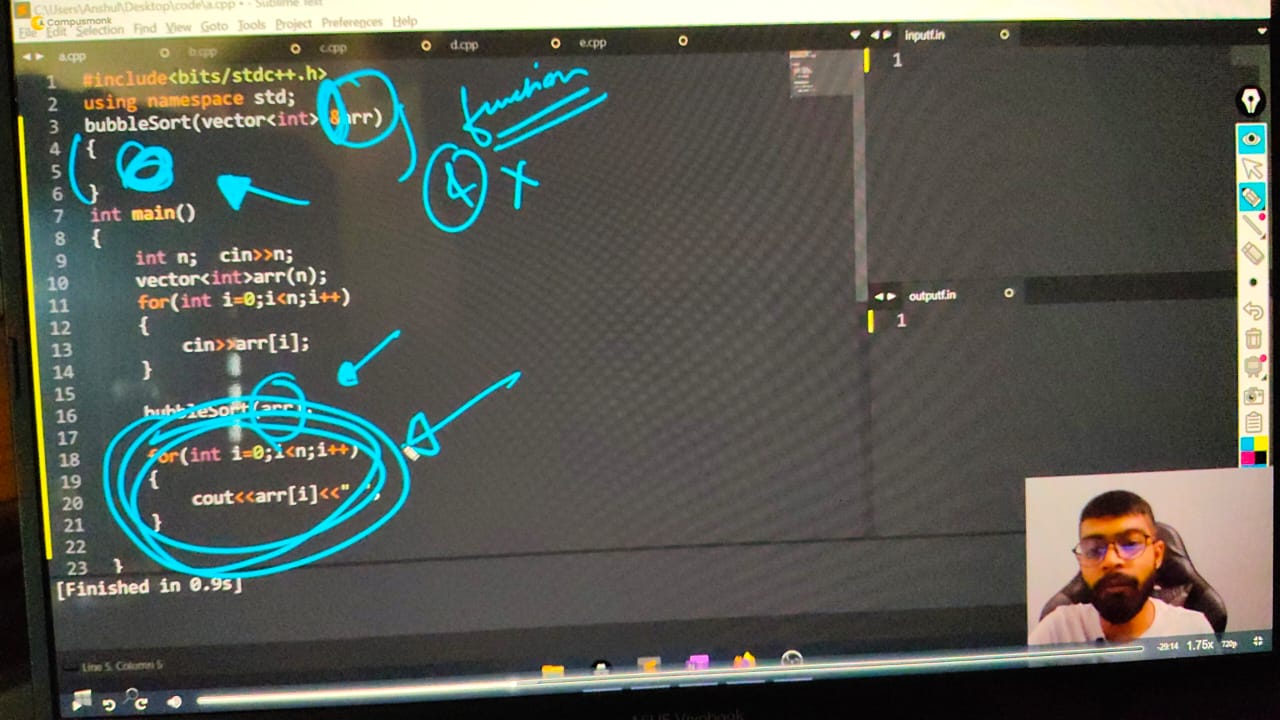
using namespace std;

**// Qun 1 - Bubble Sort Implementation**

// void bubbleSort(vector<int> &arr) // calling througb call by address - mtlb yaha change hua to niche bhi array me change hona chahiye







// {

//     int n = arr.size();

//     for (int i = 0; i < n; i++) // vo loop jo saare elemenets ke liye chlega

//     {

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

//         {

//             // agar agli value chhoti  he to swipe kre

//             if (arr[j] > arr[j + 1])

//             {

//                 swap(arr[j + 1], arr[j]);

//             }

//         }

//     }

// }

// int main()

// {

//     int n;

//     cout << "Size of the vector - " << endl;

//     cin >> n;

//     vector<int> arr(n);

//     cout << "Enter the Vector Elements " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << endl;

//     cout << "Hence the sorted arrau is - " << endl;

//     bubbleSort(arr);

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

// /\*

// Size of the vector -

// 5

// Enter the Vector Elements

// 333 1 3208 33 69

// Hence the sorted arrau is -

// 1 33 69 333 3208

// \*/

// }

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

**// Qun 1.1 - Bubble Sort Implementation for decrasing sorting**

//  void bubbleSort(vector<int> &arr) // calling througb call by address - mtlb yaha change hua to niche bhi array me change hona chahiye

// {

//     int n = arr.size();

//     for (int i = 0; i < n; i++) // vo loop jo saare elemenets ke liye chlega

//     {

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

//         {

//             // agar agli value chhoti  he to swipe kre

//             if (arr[j] > arr[j + 1])

//             {

//                 swap(arr[j + 1], arr[j]);

//             }

//         }

//     }

// }

// int main()

// {

//     int n;

//     cout << "Size of the vector - " << endl;

//     cin >> n;

//     vector<int> arr(n);

//     cout << "Enter the Vector Elements " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << endl;

//     cout << "Hence the sorted arrau is - " << endl;

//     bubbleSort(arr);

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

// }

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

// W/o uisng function -

// int main()

// {

//     int n;

//     cout << "Vector Size" << endl;

//     cin >> n;

//     vector<int> arr(n);

//     cout << "Values of Vector Elemnets" << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     // trying bubble sort without using function

//     for (int i = 0; i < n; i++)

//     {

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

//         {

//             if (arr[j] > arr[j + 1])

//             {

//                 swap(arr[j], arr[j + 1]);

//             }

//         }

//     }

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

//     for(int i: arr)

//     {

//         cout<<i<<" ";

//     }

// /\*

// Vector Size

// 6

// Values of Vector Elemnets

// 56 89 10 5 12 36

// So, the new sorted array is -

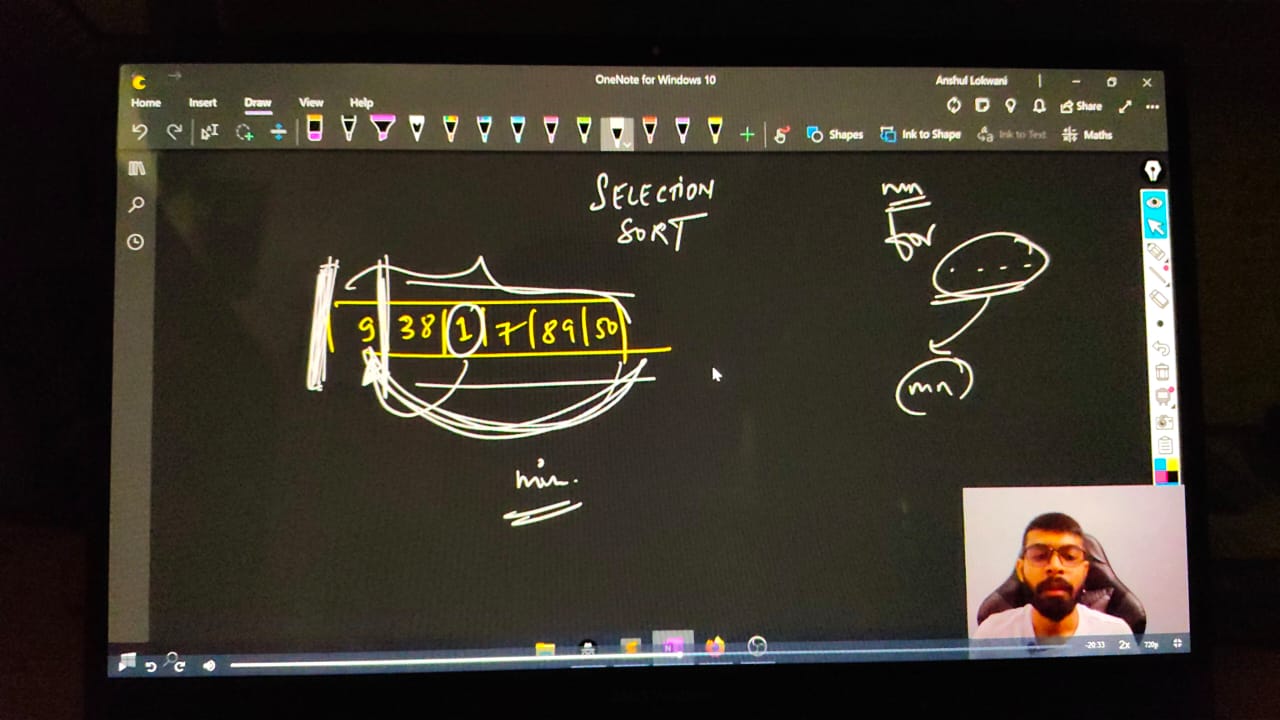
// 5 10 12 36 56 89

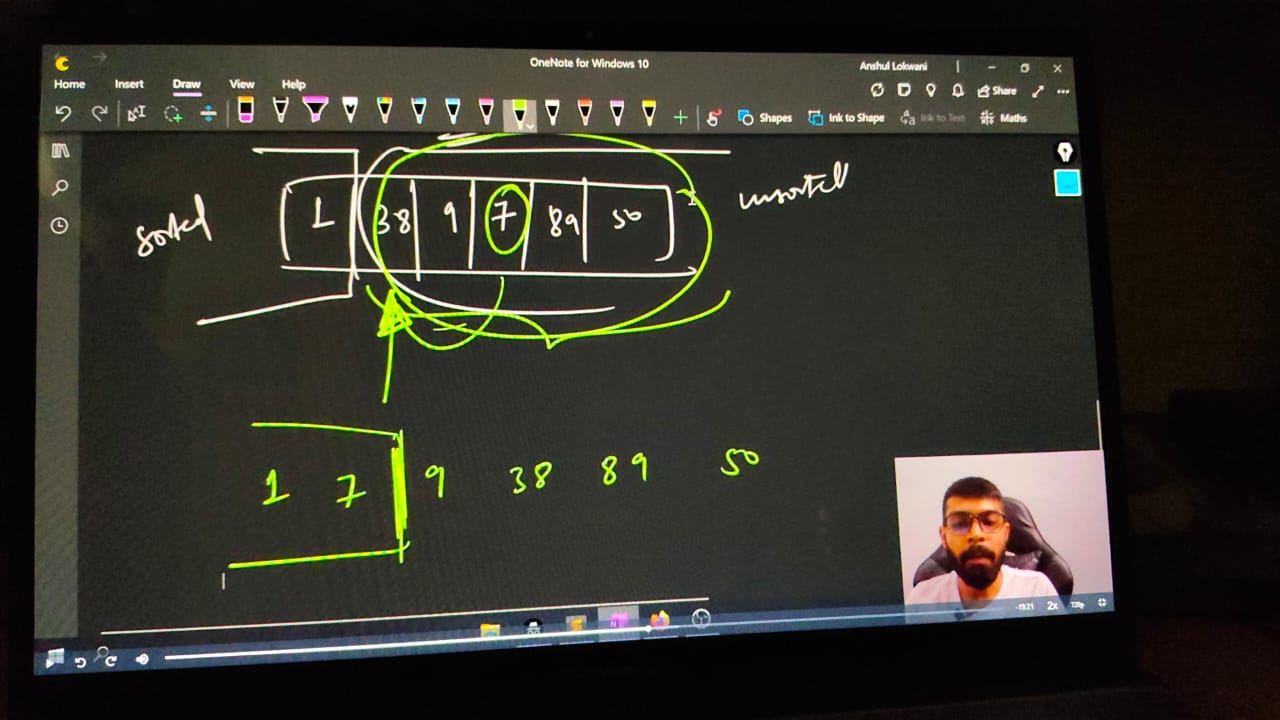
// \*/

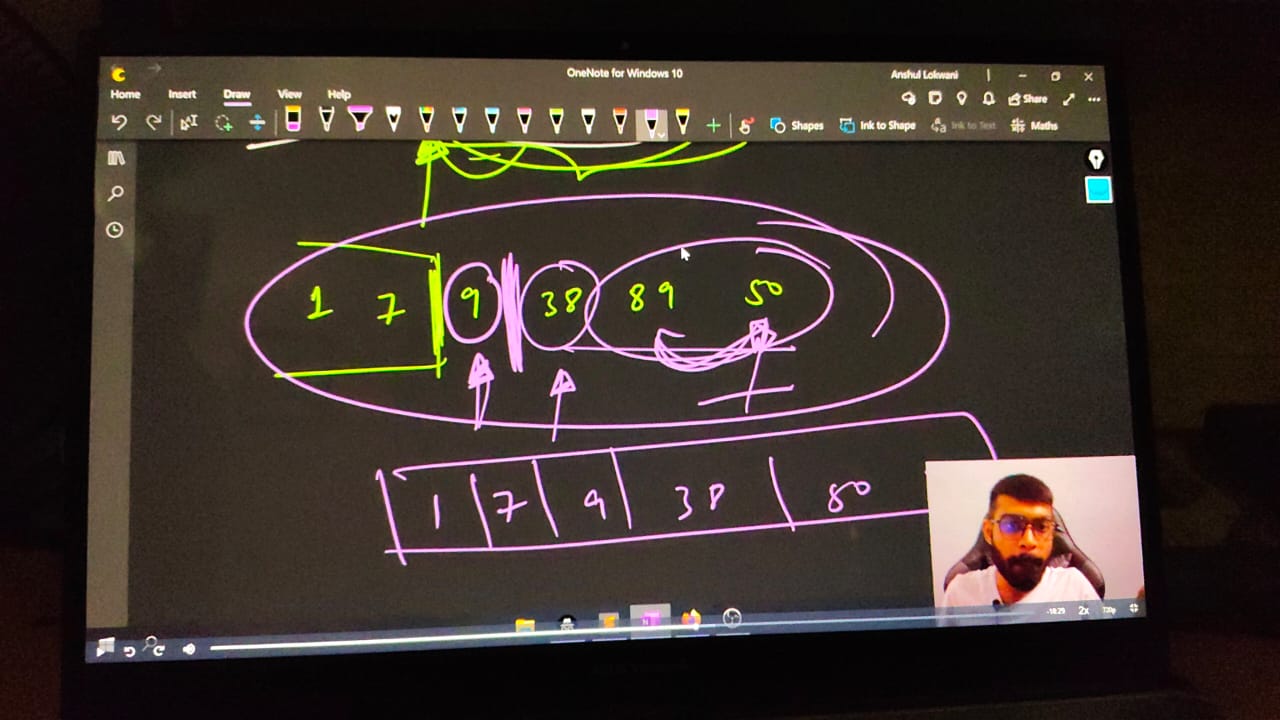
// }

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

**// Qun 2 - Selection Sort Implementaiton -**







// void insertSort(vector<int> &arr)

// {

//     int n = arr.size();

//     // insertion sort

//     for (int i = 0; i < n; i++)

//     {

//         int min = INT\_MAX;

//         int minindx = -1;

//         // min value

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

//         {

//             if (arr[j] < min)

//             {

//                 min = arr[j];

//                 minindx = j;

//             }

//         }

//         // Swapping minimum element from unsorted starting index

//         swap(arr[i], arr[minindx]); // pure me se min dhundh k use first waaali value se swap kr do

//     }

// }

// int main()

// {

//     int n;

//     cout << "Vector Size" << endl;

//     cin >> n;

//     vector<int> arr(n);

//     cout << "Mention the elements" << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     insertSort(arr);

//     cout << "So, the sorted array using Selection Sort is - " << endl;

//     for (int i : arr)

//     {

//         cout << i << " ";

//     }

//     /\*

//     Vector Size

//     6

//     Mention the elements

//     99 2 11 2222 64 1726

//     So, the sorted array using Insertion Sort is -

//     2 11 64 99 1726 2222

//     \*/

// }

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

**// 2.1 -Implement Selection Sort for Decreasing Order**

// void insertSort(vector<int> &arr)

//  {

//      int n = arr.size();

//      // insertion sort

//     for (int i = 0; i < n; i++)

//     {

//         int max = INT\_MIN;

//         int maxindx = -1;

//         // Max value

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

//         {

//             if (arr[j] > max)

//             {

//                 max = arr[j];

//                 maxindx = j;

//             }

//         }

//         // Swapping maximum element from unsorted starting index

//         swap(arr[i], arr[maxindx]); // pure me se max dhundh k use first waaali value se swap kr do

//     }

// }

// int main()

// {

//     int n;

//     cout << "Vector Size" << endl;

//     cin >> n;

//     vector<int> arr(n);

//     cout << "Mention the elements" << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     insertSort(arr);

//     cout << "So, the sorted array using Selection Sort is - " << endl;

//     for (int i : arr)

//     {

//         cout << i << " ";

//     }

/\*

Vector Size

6

Mention the elements

99 2 11 2222 64 1726

So, the sorted array using Insertion Sort is -

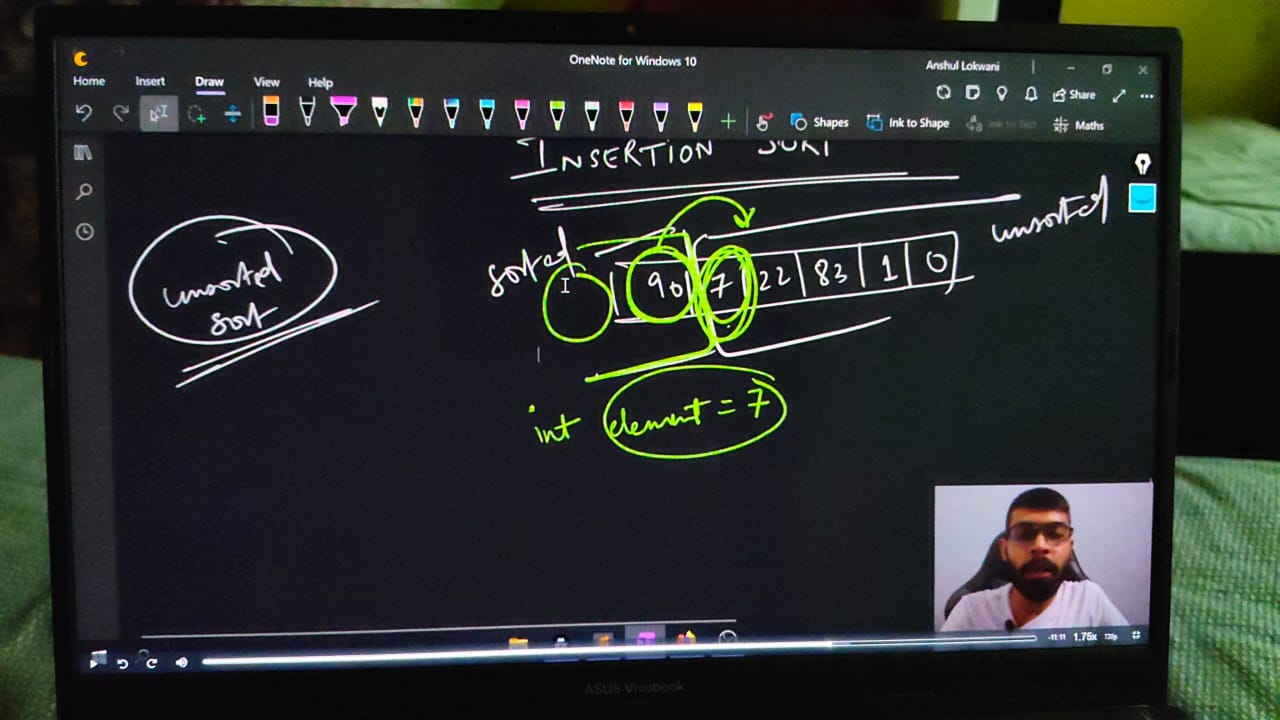
2222 1726 99 64 11 2

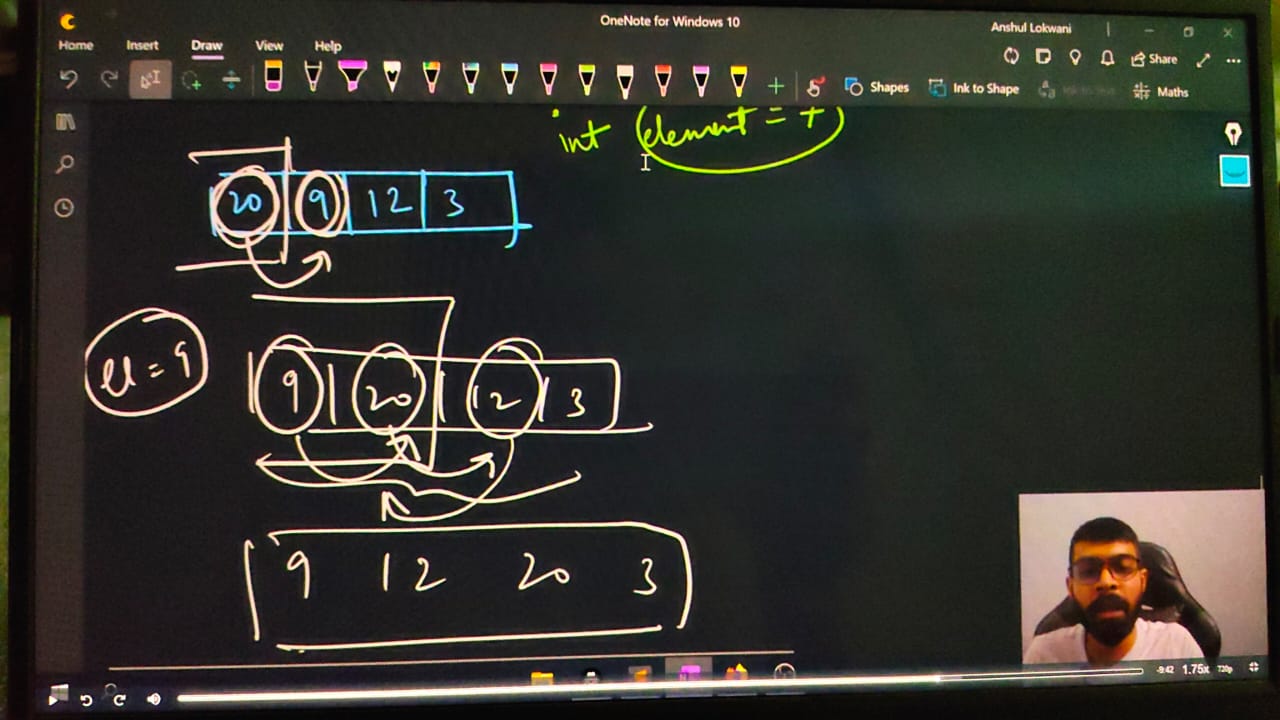
\*/

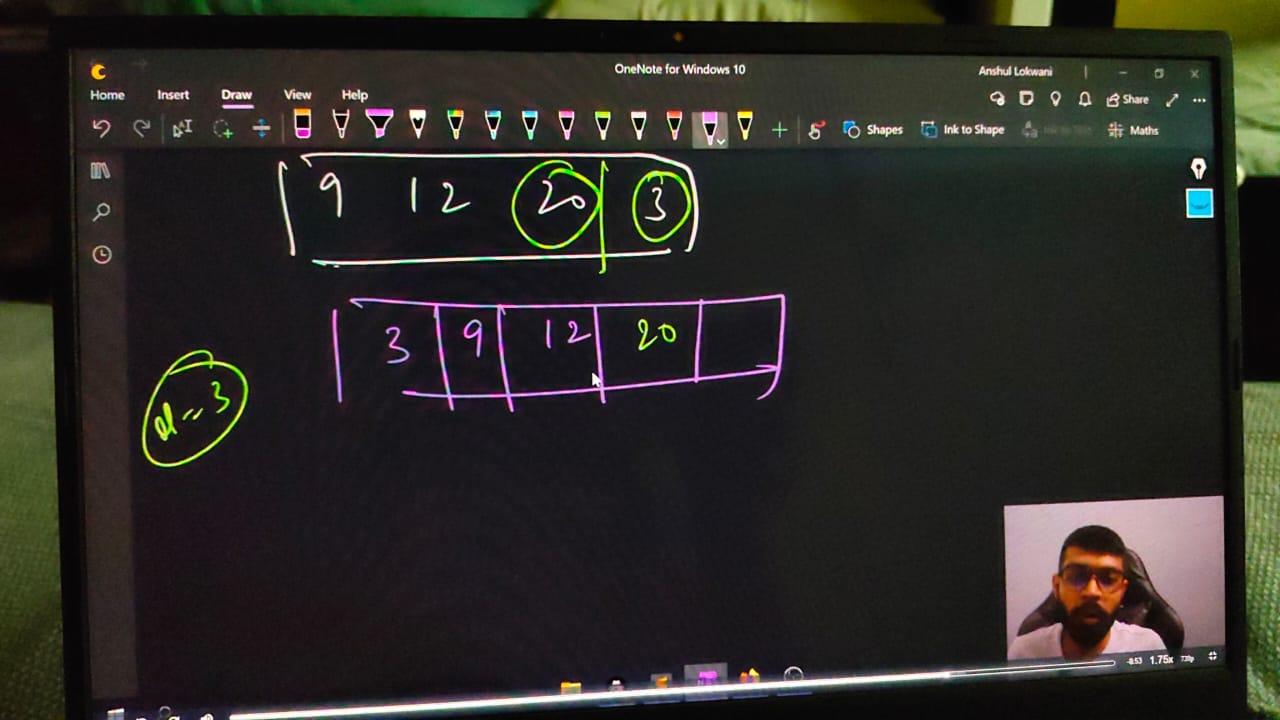
//}

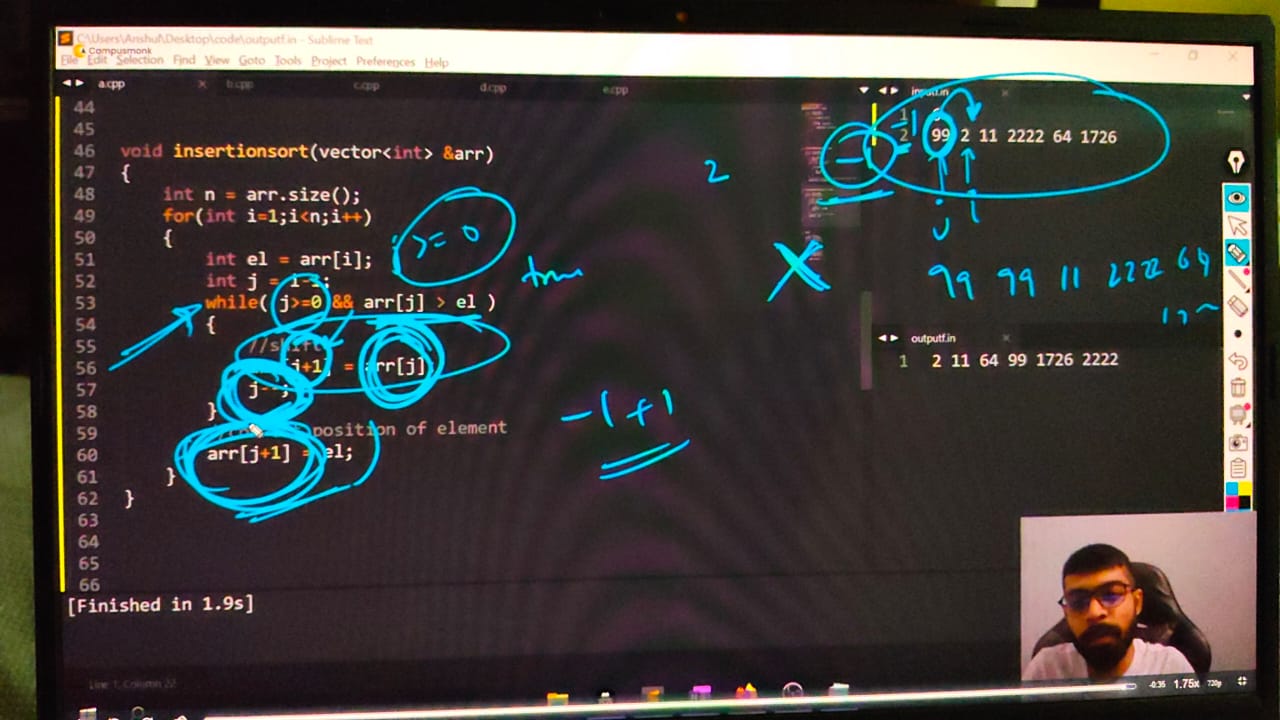
// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// Qun 3 - Insertion Sort Implementation –**









// void insertionSort(vector<int> &arr)

// {

//     int n = arr.size();

//     for (int i = 1; i < n; i++)

//     {

//         int el = arr[i];

//         int j = i - 1;

//         while (j >= 0 && arr[j] > el)

//         {

//             // Shift

//             arr[j + 1] = arr[j];

//             j--;

//         }

//         // Correct possition of element -

//         arr[j+1]=el;

//     }

// }

// int main()

// {

//     int n;

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

//     cin >> n;

//     vector<int> arr(n);

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

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     insertionSort(arr);

//     cout << "So, the sorted array after applying insertion sort is -" << endl;

//     for (int i : arr)

//     {

//         cout << i << " ";

//     }

// /\*

// Mention the Vector size

// 6

// Enter the vector elements

// 99 2 11 2222 64 1726

// So, the sorted array after applying insertion sort is -

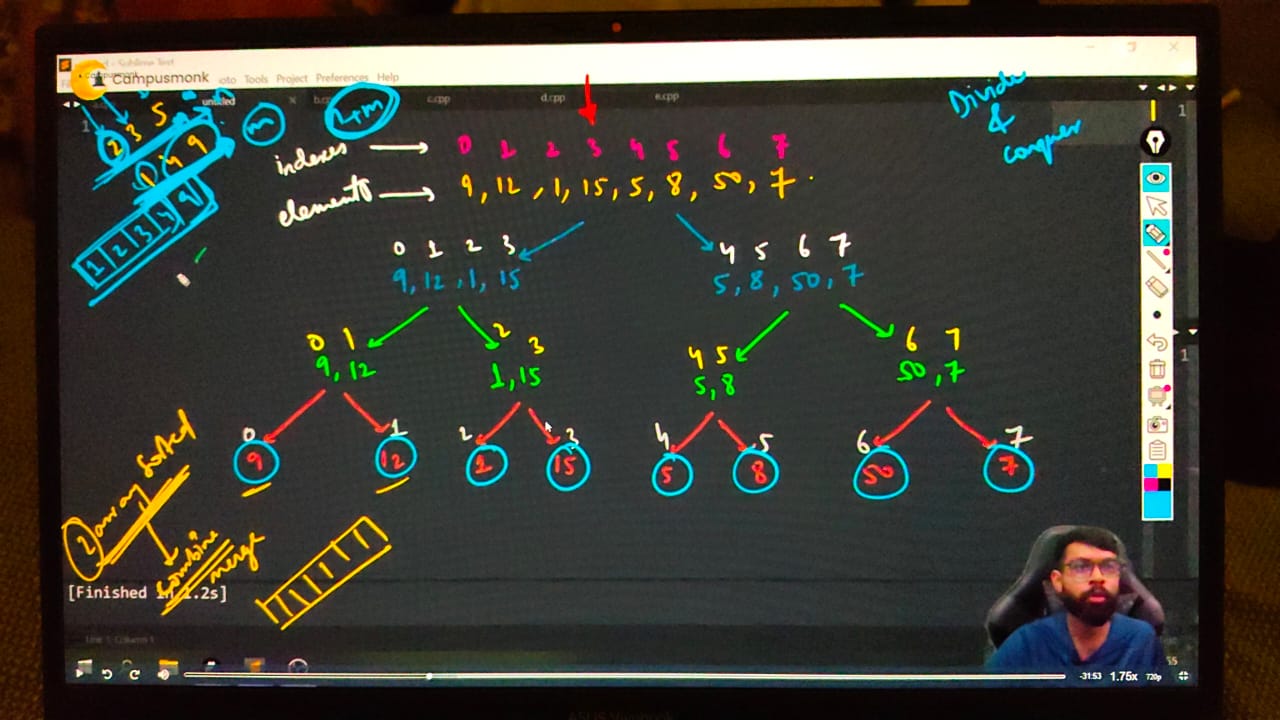
// 2 11 64 99 1726 2222

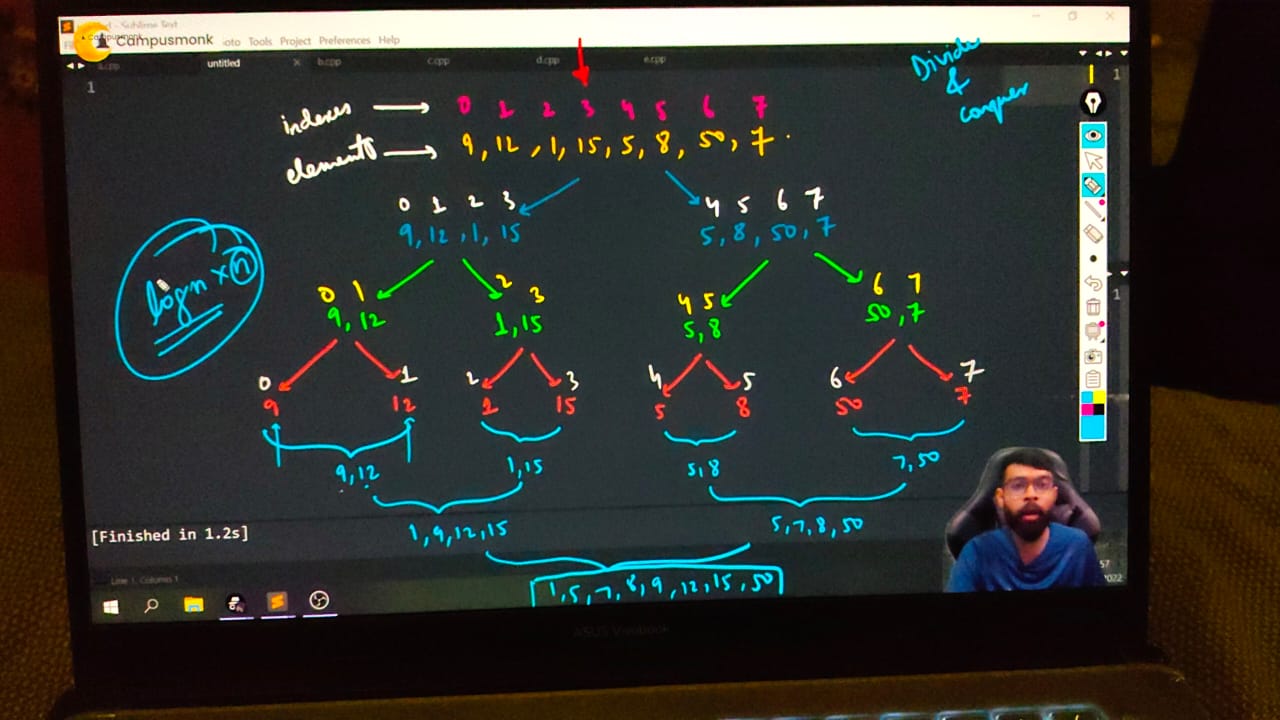
// \*/

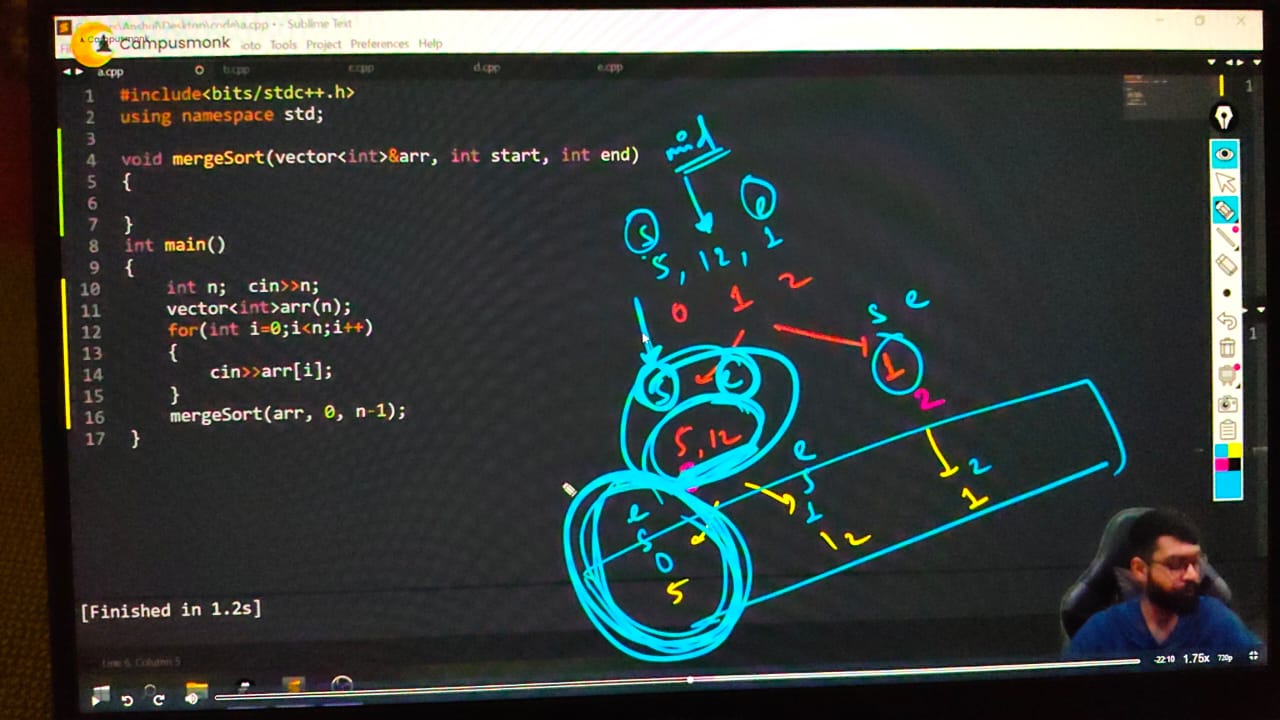
// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

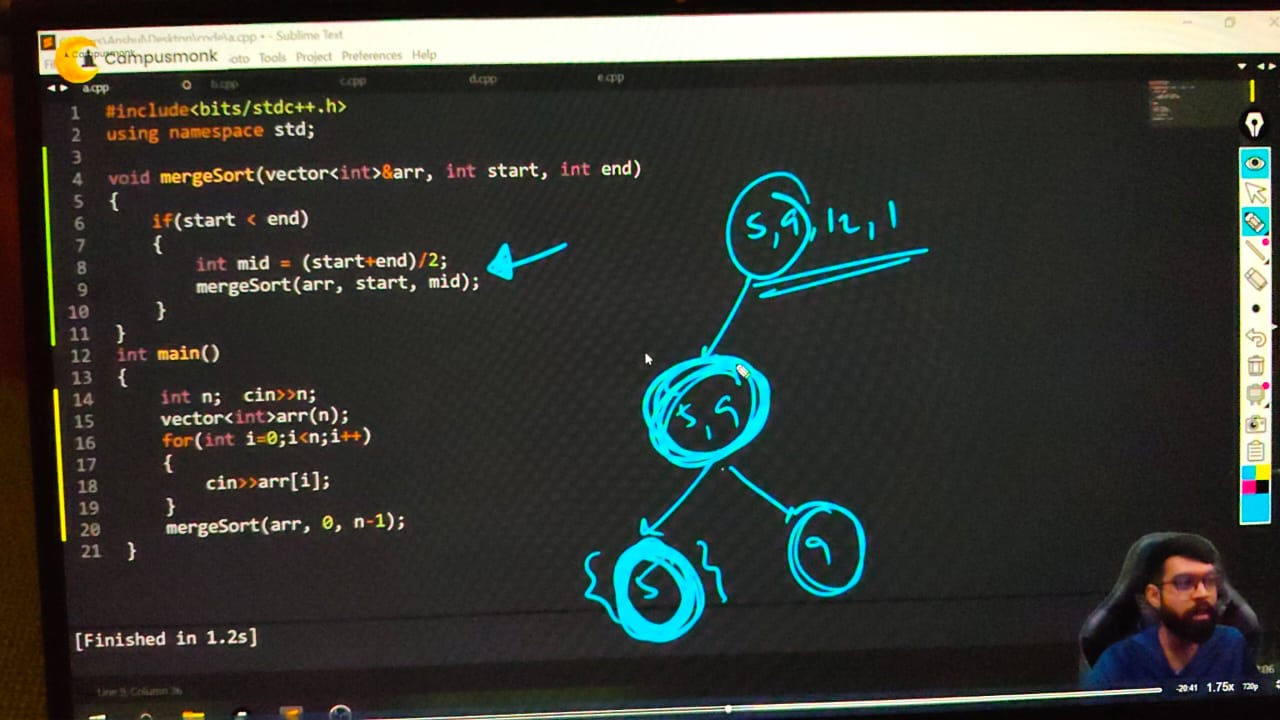
}

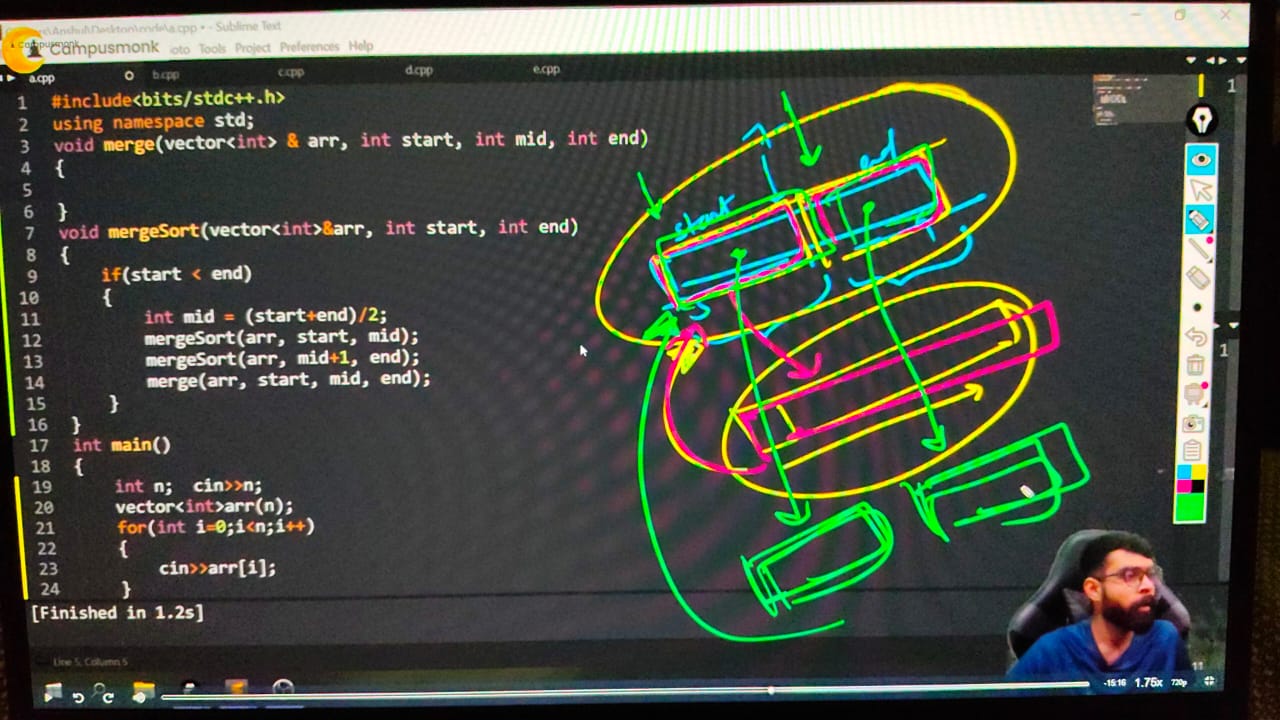
**//Qun 4 - Merge Sort Implementation**

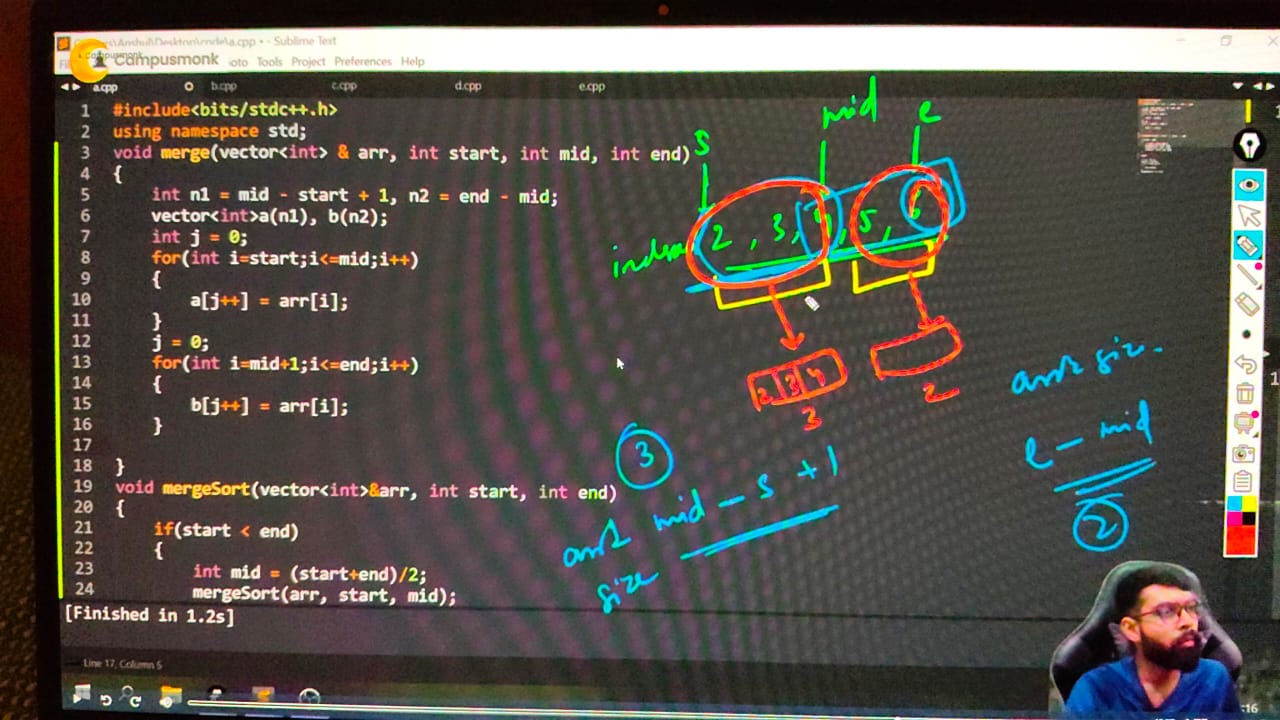


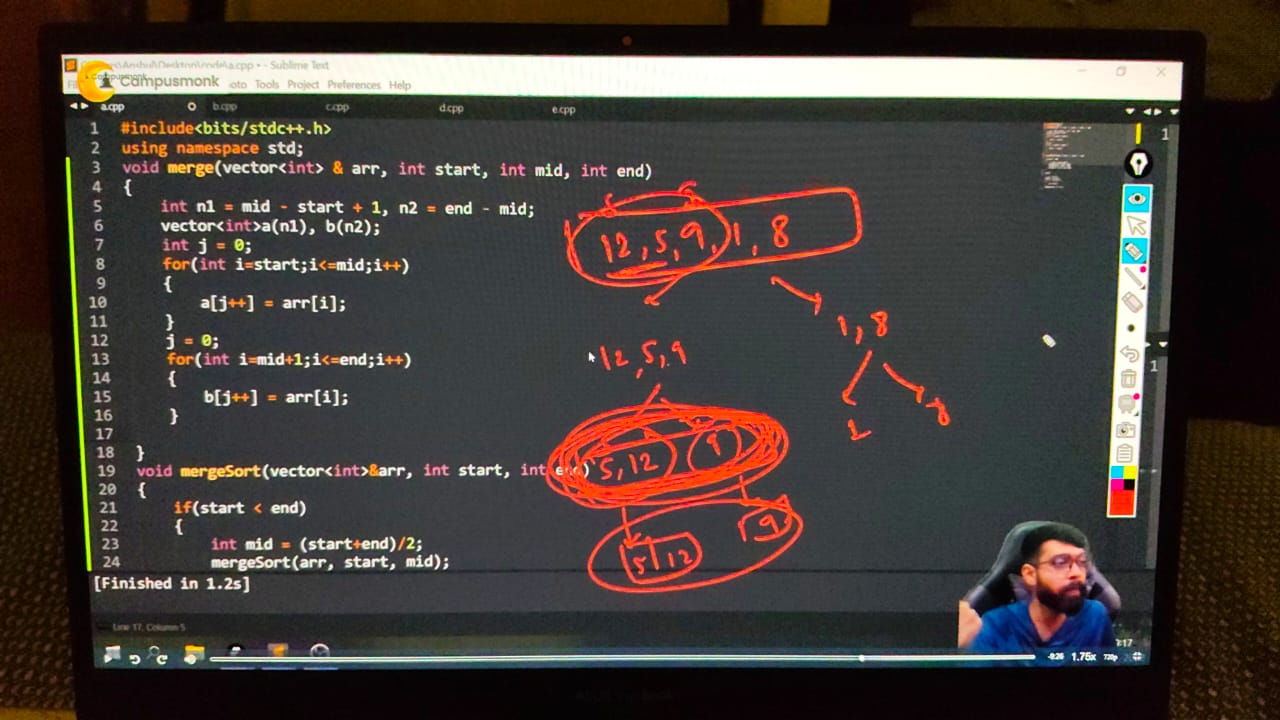


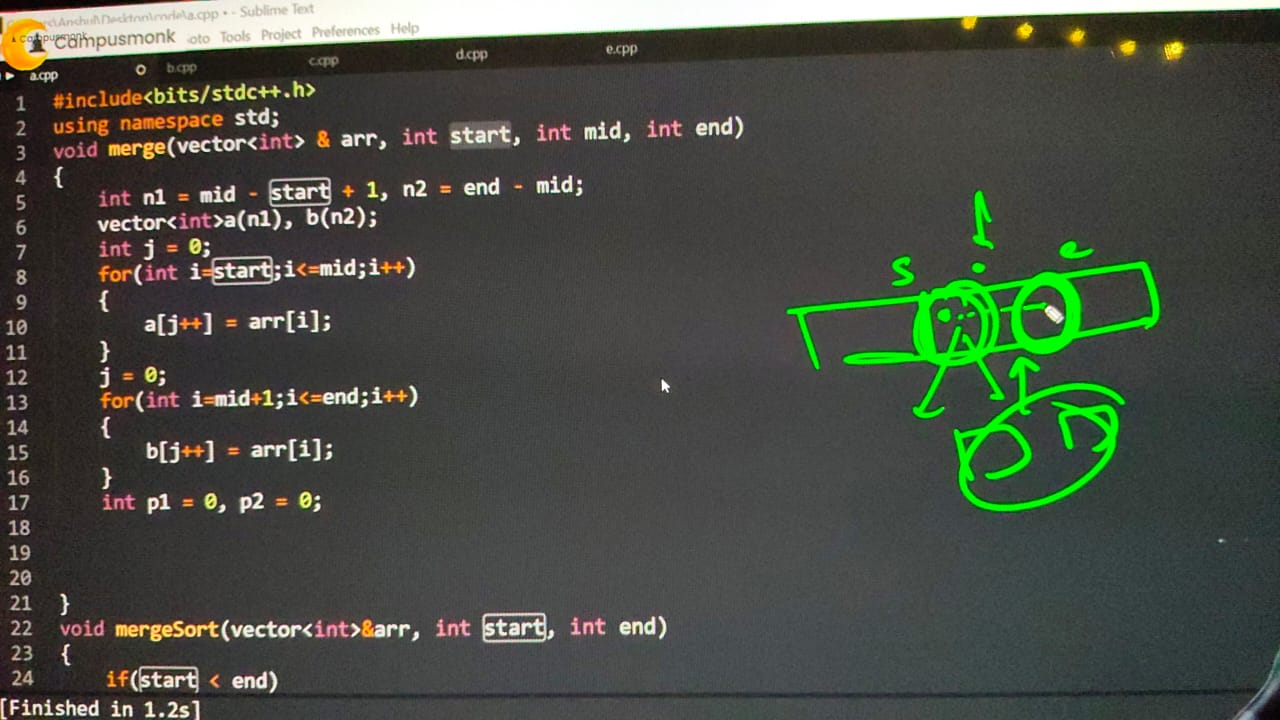












/\*

In Insertion, Bubble & Selection All three sorting algo having T.C = O(n^2) in worst case & S.C - is O(1)

But in Merge case in WC, T.C - is always O(n\*logn) & SC is  - O(1)

 \*/

// void merge(vector<int> &arr, int start, int mid, int end)

// {

//     int n1 = mid - start + 1, n2 = end - mid;

//     vector<int> a(n1), b(n2);

//     int j = 0;

//     for (int i = start; i <= mid; i++)

//     {

//         a[j++] = arr[i];

//     }

//     j = 0;

//     for (int i = mid + 1; i <= end; i++)

//     {

//         b[j++] = arr[i];

//     }

//     int p1 = 0, p2 = 0;

//     int originalPointer = start;

//     while (p1 < n1 && p2 < n2)

//     {

//         if (a[p1] < b[p2])

//         {

//             arr[originalPointer++] = a[p1++];

//         }

//         else

//         {

//             arr[originalPointer++] = b[p2++];

//         }

//     }

// }

// void mergeSort(vector<int> &arr, int start, int end)

// {

//     if (start < end)

//     {

//         int mid = (start + end) / 2;

//         mergeSort(arr, start, mid);

//         mergeSort(arr, mid + 1, end);

//         merge(arr, start, mid, end);

//     }

// }

// int main()

// {

//     int n;

//     cout << "Mention the sixe of Vector - " << endl;

//     cin >> n;

//     vector<int> arr(n);

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

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     mergeSort(arr, 0, n - 1);

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

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

/\*

ention the sixe of Vector -

6

Enter the elements of vector -

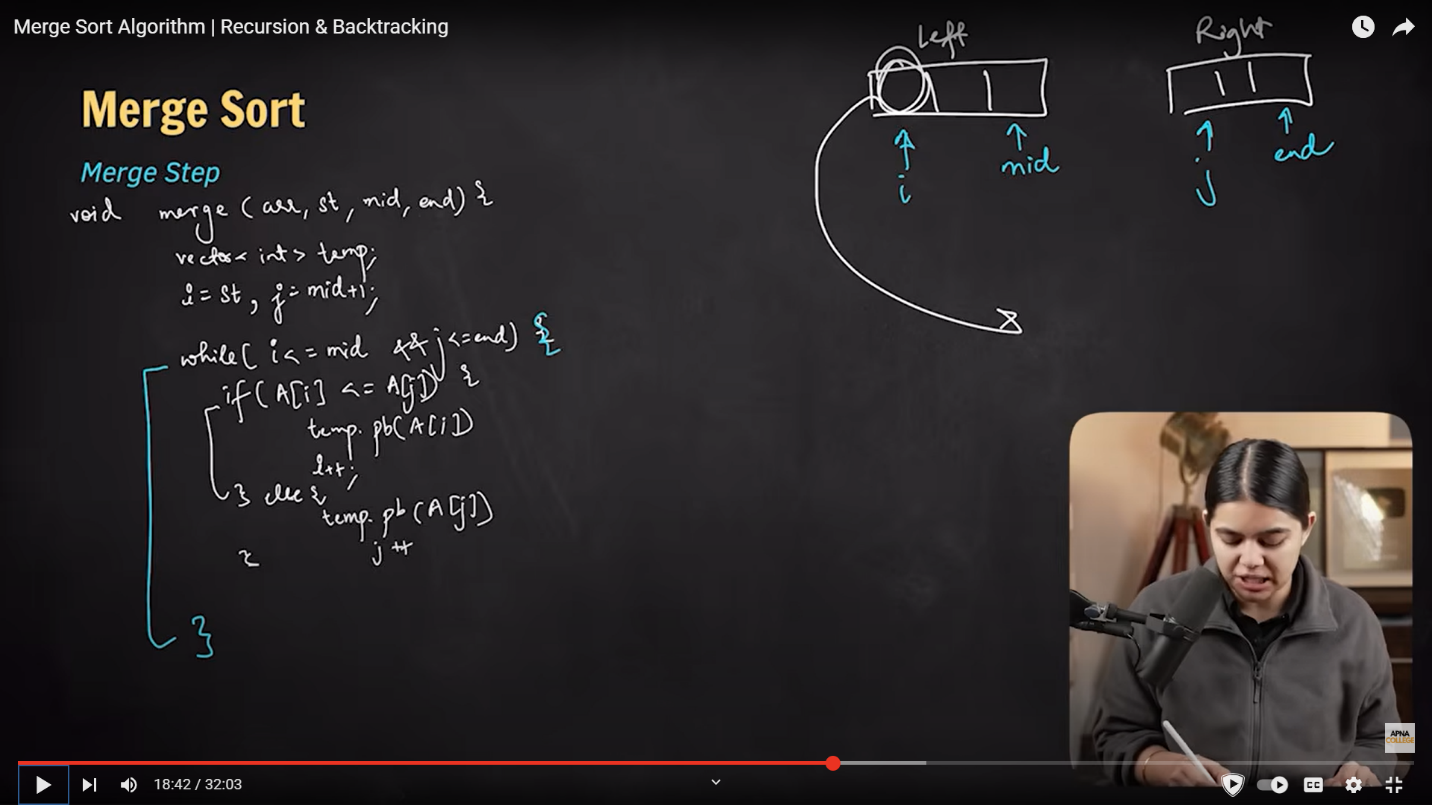
30 2 15 44 6 8

So, the array after sorting  is -

2 2 6 6 8 8

 \*/

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



**// Merge Sort using different method -**

**// Function to merge two sorted halves of the array**

// void merge(vector<int> &arr, int st, int mid, int end) {

//     vector<int> temp;

//     int i = st, j = mid + 1;

//     // Merge the two sorted halves

//     while (i <= mid && j <= end) {

//         if (arr[i] <= arr[j]) {

//             temp.push\_back(arr[i]);

//             i++;

//         } else {

//             temp.push\_back(arr[j]);

//             j++;

//         }

//     }

//     // Add remaining elements from the left half

//     while (i <= mid) {

//         temp.push\_back(arr[i]);

//         i++;

//     }

//     // Add remaining elements from the right half

//     while (j <= end) {

//         temp.push\_back(arr[j]);

//         j++;

//     }

//     // Copy back the sorted elements into original array

//     for (int idx = 0; idx < temp.size(); idx++) {

//         arr[st + idx] = temp[idx];

//     }

// }

// // Merge sort function

// void mergeSort(vector<int> &arr, int st, int end) {

//     if (st < end) {

//         int mid = st + (end - st) / 2;

//         mergeSort(arr, st, mid);       // Left half

//         mergeSort(arr, mid + 1, end);  // Right half

//         merge(arr, st, mid, end);      // Merge both halves

//     }

// }

// int main() {

//     vector<int> arr = {12, 31, 35, 8, 32, 17};

//     mergeSort(arr, 0, arr.size() - 1);

//     cout << "Sorted array: ";

//     for (int val : arr) {

//         cout << val << " ";

//     }

//     cout << endl;

//     return 0;

// // Sorted array: 8 12 17 31 32 35

// }

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

// }