|  |
| --- |
| Practical no :-01  Aim : Write a program in C++ to implement Array.  Solution :-  #include <iostream>  using namespace std;  int main(){     float percentage[] = {56.4 , 99.0, 12.20, 67.2};     cout<<"printing all values of the array :\n";     for(int i = 0; i<4 ; i++){        cout<<"element "<<i+1<<" = "<<percentage[i]<<endl;     }     return 0;  }  Output :-  Practical No :- 02  Aim : Write a program to accept the elements in 2D array and perform |
| all the matrix operations i.e. addition, multiplication, substraction etc. |

A)

Solution :-

#include <iostream>

using namespace std;

int main() {

   int r=2, c=4, sum[2][4], i, j;

   int a[2][4] = {{1,5,9,4} , {3,2,8,3}};

   int b[2][4] = {{6,3,8,2} , {1,5,2,9}};

   cout<<"The first matrix is: "<<endl;

   for(i=0; i<r; ++i) {

      for(j=0; j<c; ++j)

      cout<<a[i][j]<<" ";

      cout<<endl;

   }

   cout<<endl;

   cout<<"The second matrix is: "<<endl;

   for(i=0; i<r; ++i) {

      for(j=0; j<c; ++j)

      cout<<b[i][j]<<" ";

      cout<<endl;

   }

   cout<<endl;

   for(i=0;i<r;++i)

   for(j=0;j<c;++j)

   sum[i][j]=a[i][j]+b[i][j];

   cout<<"Sum of the two matrices is:"<<endl;

   for(i=0; i<r; ++i) {

      for(j=0; j<c; ++j)

      cout<<sum[i][j]<<" ";

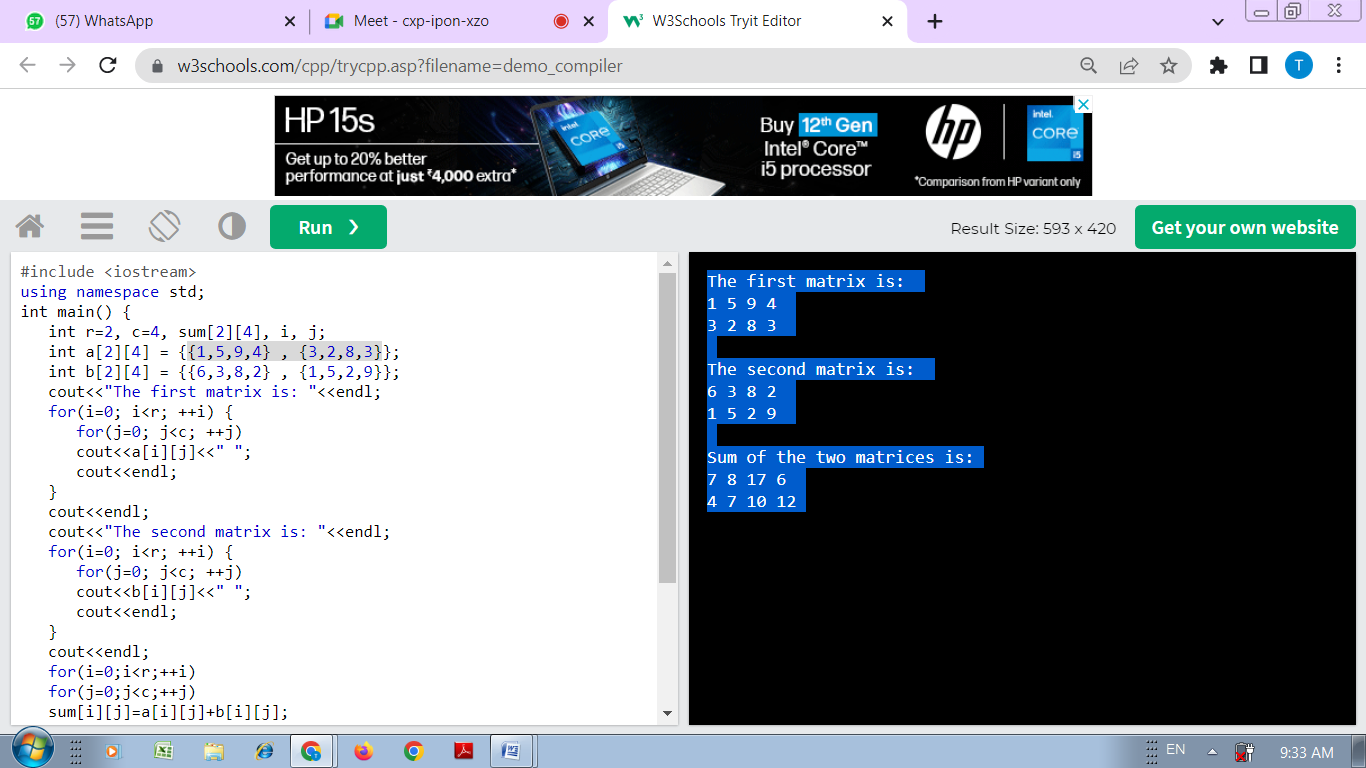
      cout<<endl;

   }

   return 0;

}

Output :-



B)

Solution :-

#include <iostream>

using namespace std;

int main() {

int r=2, c=4, sub[2][4], i, j;

int a[2][4] = {{1,5,9,4} , {3,2,8,3}};

int b[2][4] = {{6,3,8,2} , {1,5,2,9}};

cout<<"The first matrix is: "<<endl;

for(i=0; i<r; ++i) {

for(j=0; j<c; ++j)

cout<<a[i][j]<<" ";

cout<<endl;

}

cout<<endl;

cout<<"The second matrix is: "<<endl;

for(i=0; i<r; ++i) {

for(j=0; j<c; ++j)

cout<<b[i][j]<<" ";

cout<<endl;

}

cout<<endl;

for(i=0;i<r;++i)

for(j=0;j<c;++j)

sub[i][j]=a[i][j]-b[i][j];

cout<<"Subtraction of the two matrices is:"<<endl;

for(i=0; i<r; ++i) {

for(j=0; j<c; ++j)

cout<<sub[i][j]<<" ";

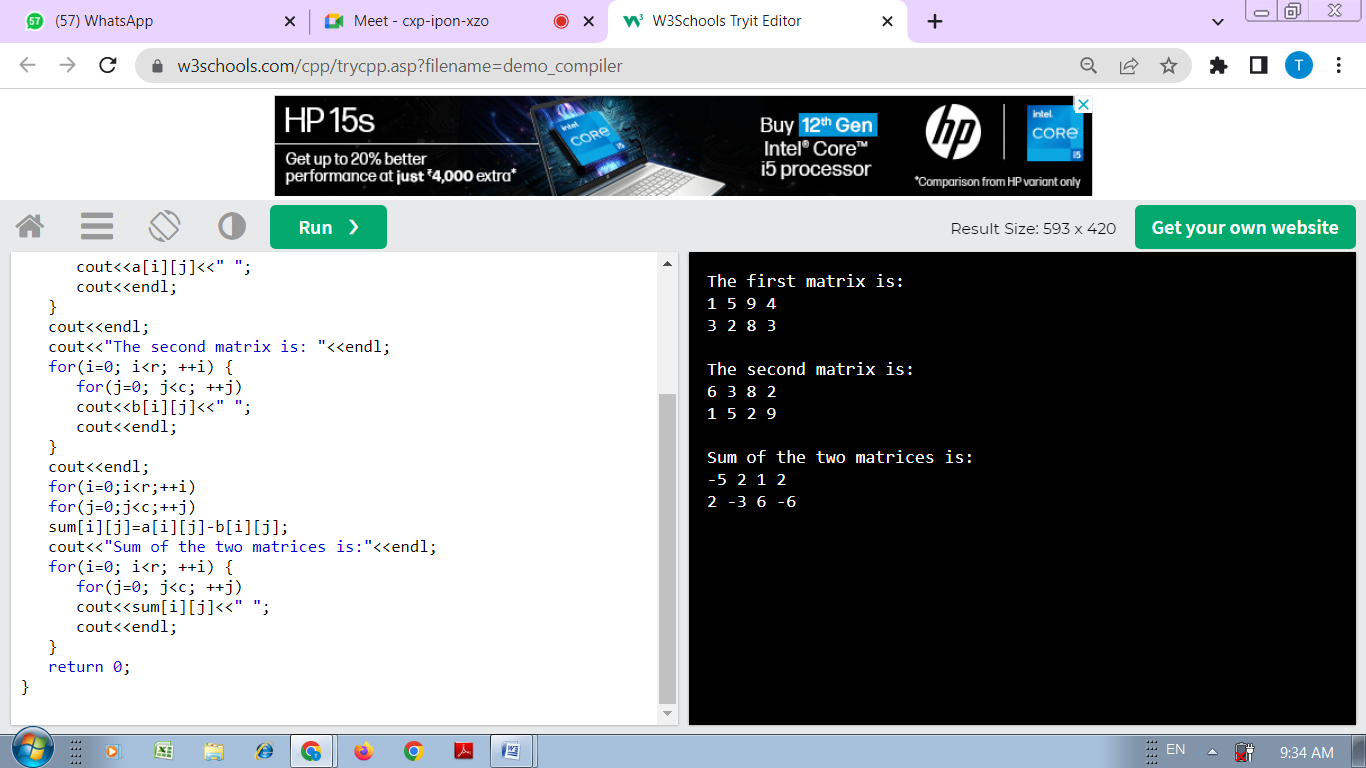
cout<<endl;

}

return 0;

}

Output :-



C)

Solution :-

#include <iostream>

using namespace std;

int main() {

int r=2, c=4, mul[2][4], i, j;

int a[2][4] = {{1,5,9,4} , {3,2,8,3}};

int b[2][4] = {{6,3,8,2} , {1,5,2,9}};

cout<<"The first matrix is: "<<endl;

for(i=0; i<r; ++i) {

for(j=0; j<c; ++j)

cout<<a[i][j]<<" ";

cout<<endl;

}

cout<<endl;

cout<<"The second matrix is: "<<endl;

for(i=0; i<r; ++i) {

for(j=0; j<c; ++j)

cout<<b[i][j]<<" ";

cout<<endl;

}

cout<<endl;

for(i=0;i<r;++i)

for(j=0;j<c;++j)

mul[i][j]=a[i][j]\*b[i][j];

cout<<"mul of the two matrices is:"<<endl;

for(i=0; i<r; ++i) {

for(j=0; j<c; ++j)

cout<<mul[i][j]<<" ";

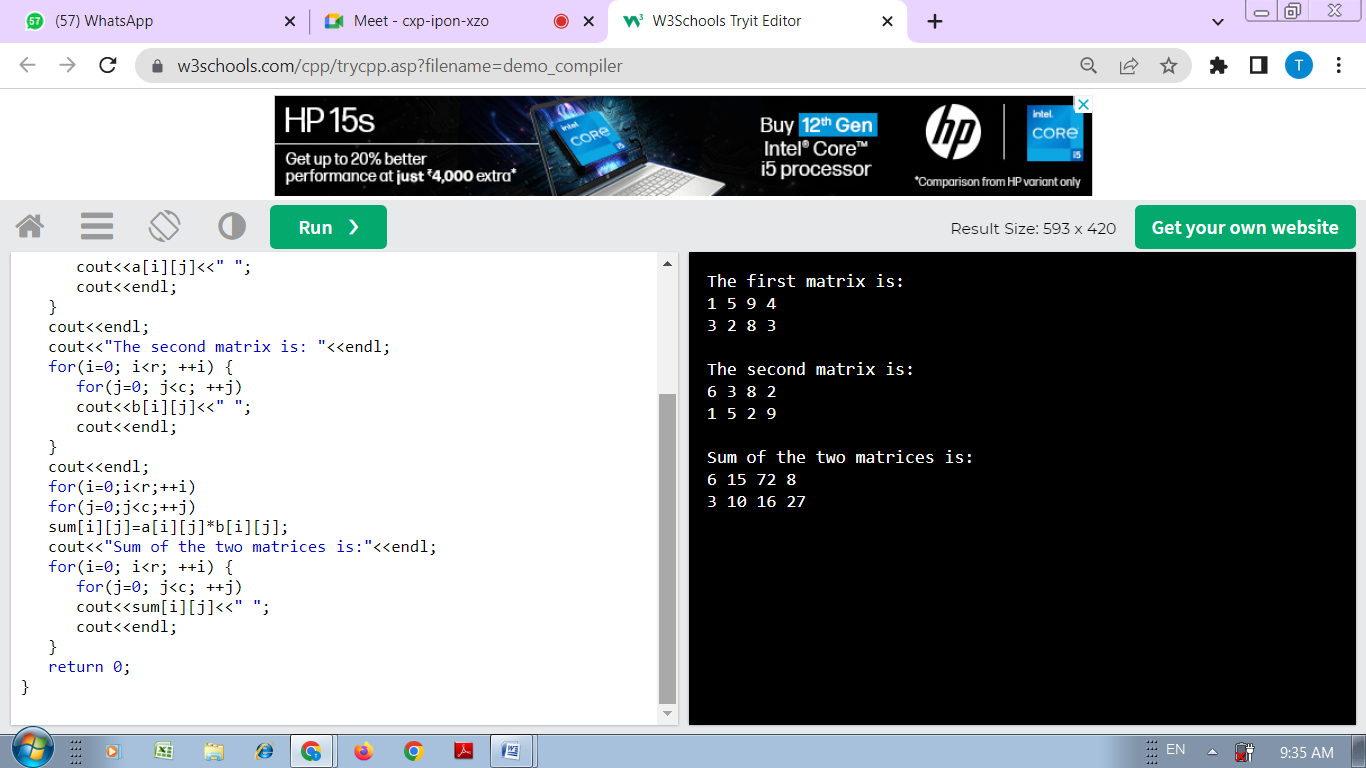
cout<<endl;

}

return 0;

}

Output :-



|  |  |  |
| --- | --- | --- |
| Practical No :-03  Aim :- Explain following techniques | | |
| Technique |  | Bubble sort |

Solution :-

// Bubble sort in C++

#include <iostream>

using namespace std;

// perform bubble sort

void bubbleSort(int array[], int size) {

// loop to access each array element

for (int step = 0; step < size; ++step) {

// loop to compare array elements

for (int i = 0; i < size - step; ++i) {

// compare two adjacent elements

// change > to < to sort in descending order

if (array[i] > array[i + 1]) {

// swapping elements if elements

// are not in the intended order

int temp = array[i];

array[i] = array[i + 1];

array[i + 1] = temp;

}

}

}

}

// print array

void printArray(int array[], int size) {

for (int i = 0; i < size; ++i) {

cout << " " << array[i];

}

cout << "\n";

}

int main() {

int data[] = {-2, 45, 0, 11, -9};

// find array's length

int size = sizeof(data) / sizeof(data[0]);

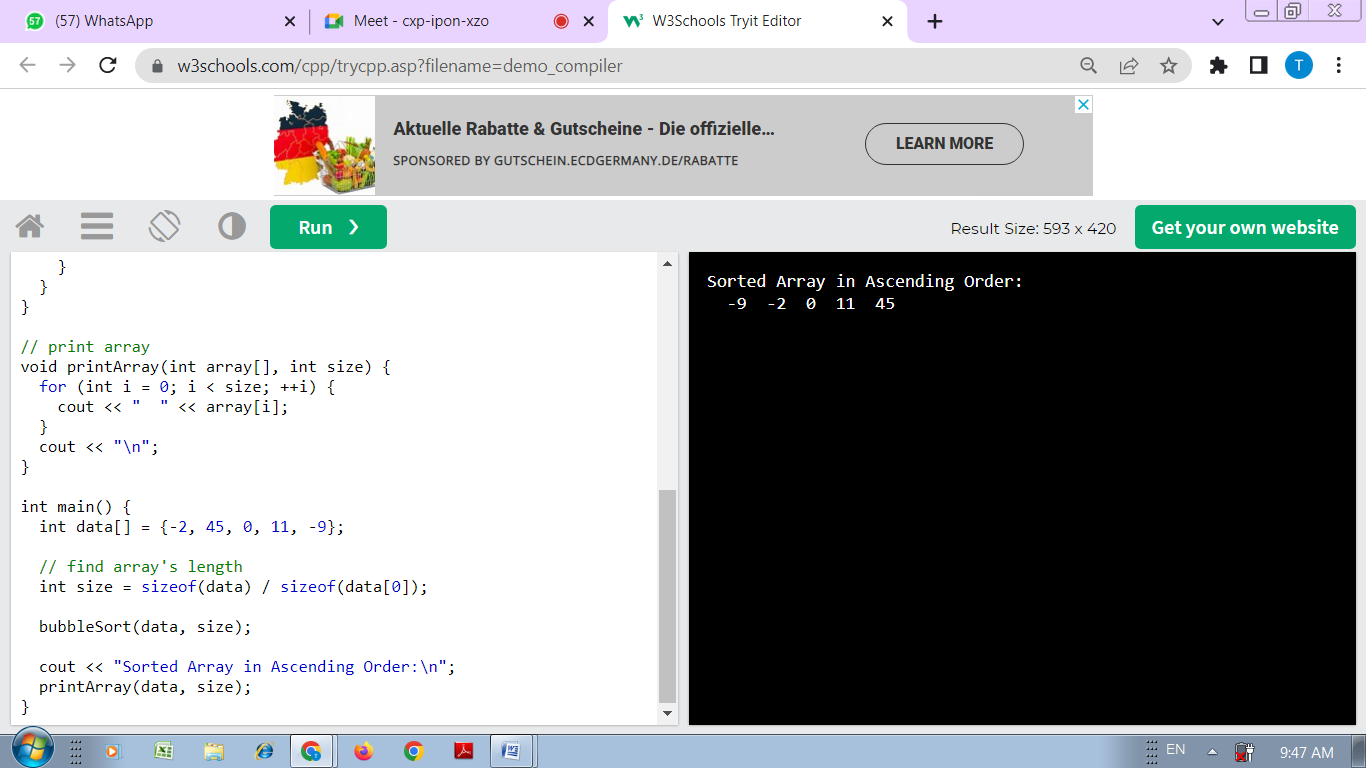
bubbleSort(data, size);

cout << "Sorted Array in Ascending Order:\n";

printArray(data, size);

}

Output :-



|  |
| --- |
| Practical No :- 04  Aim : Write a program in C++ to delete particular element from an array of 10 integers. |
|  |

Solution :-

#include<iostream>

using namespace std;

int main()

{

int arr[10], tot=10, i, elem, j, found=0;

cout<<"Enter 10 Array Elements: ";

for(i=0; i<tot; i++)

cin>>arr[i];

cout<<"\nEnter Element to Delete: ";

cin>>elem;

for(i=0; i<tot; i++)

{

if(arr[i]==elem)

{

for(j=i; j<(tot-1); j++)

arr[j] = arr[j+1];

found++;

i--;

tot--;

}

}

if(found==0)

cout<<"\nElement doesn't found in the Array!";

else

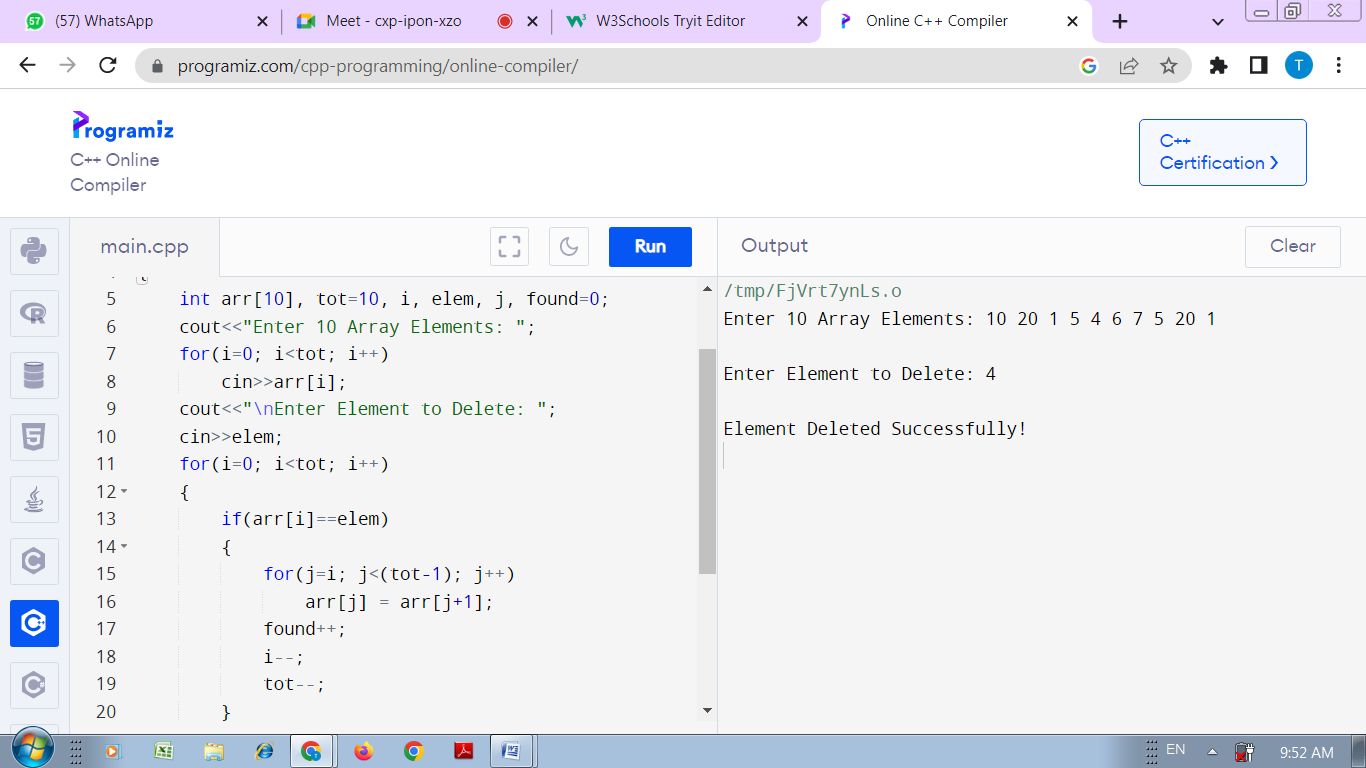
cout<<"\nElement Deleted Successfully!";

cout<<endl;

return 0;

}

Output :-



|  |
| --- |
| Practical No :- 05  Aim :-  Consider two single dimensional array of size 20 and 3 |
| respectively. Write a program in C++ to display all the elements which are common in both arrays. |
|  |

Solution :-

#include <bits/stdc++.h>

using namespace std;

int main(){

   //defining the array

   int arr1[] = { 1, 45, 54, 71, 76, 12 };

   int arr2[] = { 1, 7, 5, 4, 6, 12 };

   int n1 = sizeof(arr1) / sizeof(arr1[0]);

   int n2 = sizeof(arr2) / sizeof(arr2[0]);

   sort(arr1, arr1 + n1);

   sort(arr2, arr2 + n2);

   cout << "First Array: ";

   for (int i = 0; i < n1; i++)

      cout << arr1[i] << " ";

   cout << endl;

   cout << "Second Array: ";

   for (int i = 0; i < n2; i++)

      cout << arr2[i] << " ";

   cout << endl;

   vector<int> v(n1 + n2);

   vector<int>::iterator it, st;

   //finding the common elements

   it = set\_intersection(arr1, arr1 + n1, arr2, arr2 + n2, v.begin());

   cout << "\nCommon elements:\n";

   for (st = v.begin(); st != it; ++st)

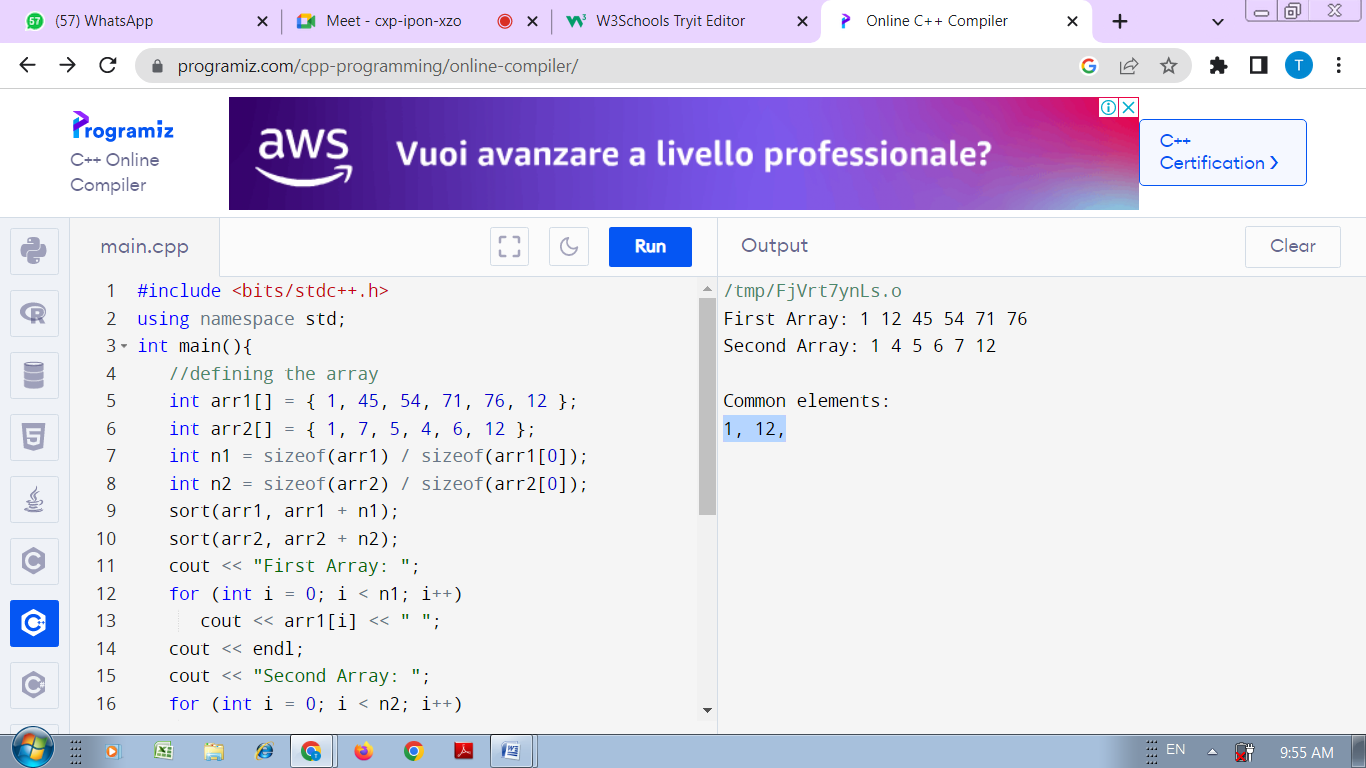
      cout << \*st << ", ";

   cout << '\n';

   return 0;

}

Output :-



Practical No :- 06

Aim :- Write a program in C++ to implement queue using Array.

Solution :-

#include <iostream>

using namespace std;

int queue[100], n = 100, front = - 1, rear = - 1;

void Insert() {

   int val;

   if (rear == n - 1)

   cout<<"Queue Overflow"<<endl;

   else {

      if (front == - 1)

      front = 0;

      cout<<"Insert the element in queue : "<<endl;

      cin>>val;

      rear++;

      queue[rear] = val;

   }

}

void Delete() {

   if (front == - 1 || front > rear) {

      cout<<"Queue Underflow ";

      return ;

   } else {

      cout<<"Element deleted from queue is : "<< queue[front] <<endl;

      front++;;

   }

}

void Display() {

   if (front == - 1)

   cout<<"Queue is empty"<<endl;

   else {

      cout<<"Queue elements are : ";

      for (int i = front; i <= rear; i++)

      cout<<queue[i]<<" ";

         cout<<endl;

   }

}

int main() {

   int ch;

   cout<<"1) Insert element to queue"<<endl;

   cout<<"2) Delete element from queue"<<endl;

   cout<<"3) Display all the elements of queue"<<endl;

   cout<<"4) Exit"<<endl;

   do {

      cout<<"Enter your choice : "<<endl;

      cin>>ch;

      switch (ch) {

         case 1: Insert();

         break;

         case 2: Delete();

         break;

         case 3: Display();

         break;

         case 4: cout<<"Exit"<<endl;

         break;

         default: cout<<"Invalid choice"<<endl;

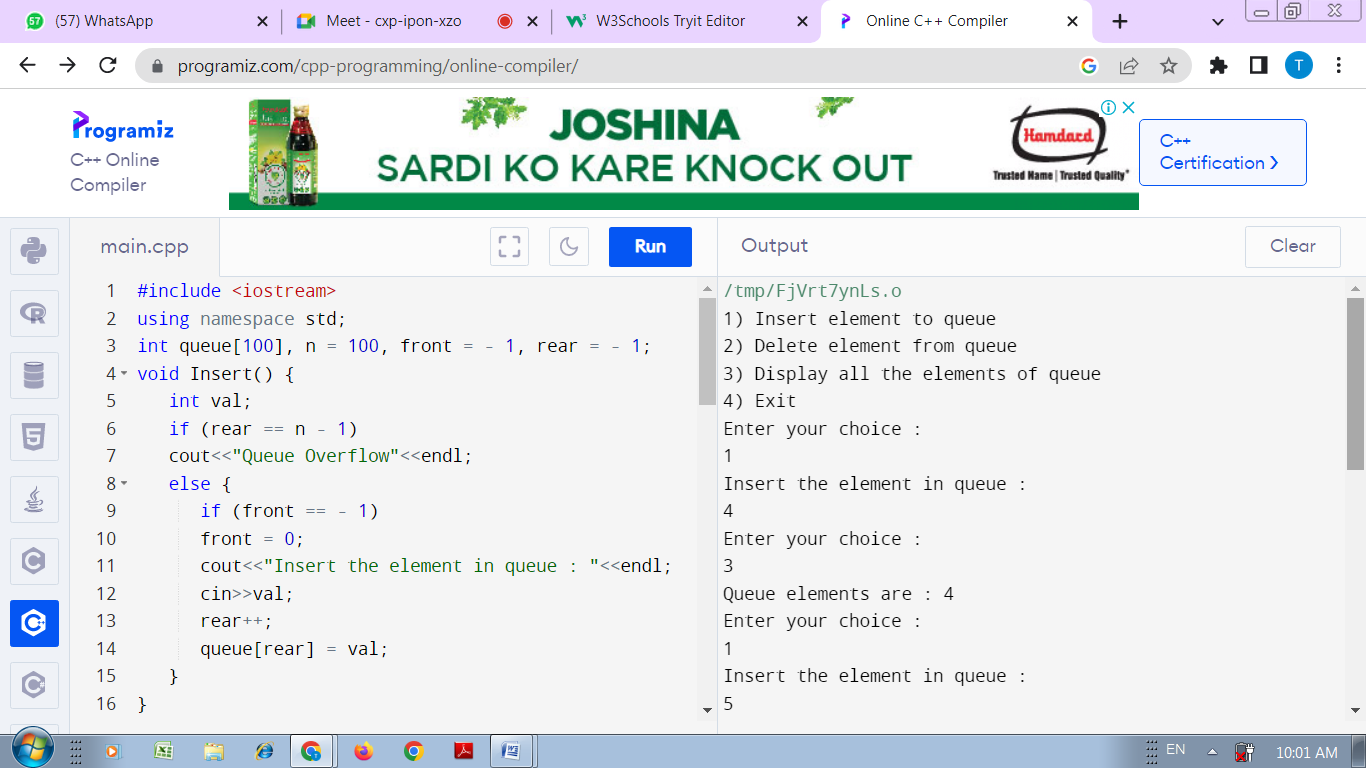
      }

   } while(ch!=4);

   return 0;

}

Output :-



|  |
| --- |
| Practical No :- 07  Aim :-  Write menu driven program which create and display the circular linked list. |
|  |

Solution :-

#include <bits/stdc++.h>

using namespace std;

struct Node

{

    int data;

    struct Node \*next;

}\*Head=NULL;

void create(int a[],int n)

{

    struct Node \*t,\*last;

    Head=new Node;

    Head->data=a[0];

    Head->next=Head;

    last=Head;

        for(int i=1;i<n;i++)

        {

           t=new Node;

    t->data=a[i];

    t->next=last->next;

    last->next=t;

    last=t;

        }

}

int count(struct Node \*p)

{

    int c=0;

    do

    {

        c++;

     p=p->next;

    }while(p!=Head);

    return c;

}

void insert(int pos, int x)

{

    if(pos>count(Head) || pos<0)

    return;

    struct Node \*t,\*p=Head;

    t=new Node;

    t->data=x;

    if(pos==0)

    {

        if(Head==NULL)

        {

            Head=t;

            Head->next=t;

        }

        while(p->next!=Head)

        p=p->next;

        t->next=Head;

        p->next=t;

        Head=t;

    }

    else{

        for(int i=0;i<pos-1;i++)

            p=p->next;

            t->next=p->next;

            p->next=t;

    }

}

int Delete(struct Node \*p,int index)

{

    struct Node \*q;

    int x=-1;

    if(index<1 || index>count(Head))

    return x;

     if(index==1)

    {

        x=p->data;

         while(p->next!=Head)

        p=p->next;

        if(Head==p)

        {

            delete Head;

            Head=NULL;

        }

        else

        {

        p->next=Head->next;

        delete Head;

        Head=p->next;

        }

        return x;

    }

    else

    {

    for(int i=0;i<index-2;i++)

    {

        p=p->next;

    }

    q=p->next;

    p->next=q->next;

    x=q->data;

    delete q;

    return x;

    }

}

void display(struct Node \*p)

{

    do

    {

        cout<<p->data<<" ";

        p=p->next;

    }while(p!=Head);

}

int main()

{

    int a[500];

    int option,n,pos,x,index,t;

    do

    {

        cout<<"1. Create Circular Linked list"<<endl<<"2. Insert in Circular Linked list"<<endl<<"3. Delete "<<endl<<"4. Display"<<endl<<"5. Exit"<<endl;;

        cout<<"Enter an option :"<<endl;

        cin>>option;

        switch(option)

        {

        case 1 :

        {

            cout<<"Enter no of integers : "<<endl;

            cin>>n;

            cout<<"Enter the numbers"<<endl;

            for(int i=0;i<n;i++)

            cin>>a[i];

            create(a,n);

            cout<<endl;

            break;

        }

        case 2:

        {

            cout<<"Enter position to insert an element : "<<endl;

            cin>>pos;

            cout<<"Enter element : "<<endl;

            cin>>x;

            insert(pos,x);

            cout<<endl;

            break;

        }

        case 3:

        {

            cout<<"Enter position to delete element : "<<endl;

            cin>>index;

            cout<<"Deleted element is: "<<Delete(Head,index);

            cout<<endl;

            break;

        }

        case 4:

        {

            cout<<"Displaying elements :";

            display(Head);

            cout<<endl;

            break;

        }

        default:

        cout<<"Exiting program......"<<endl;

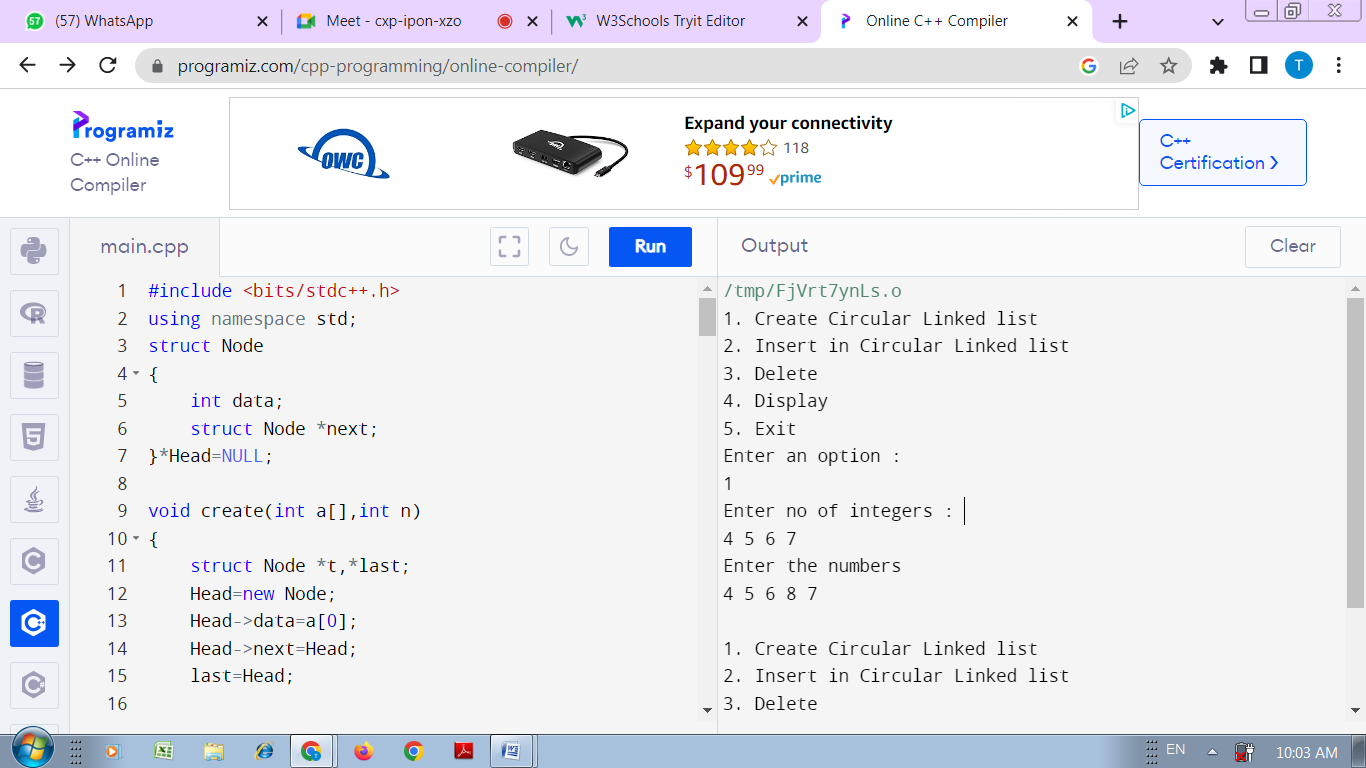
        }

    }while(option<=4);

    return 0;

}

Output :-



Practical No :- 08

Aim :- Create binary search tree 15, 2, 25, 45, 35, 23, 100, 5

Solution :-

/\* A O(n) program for construction of

BST from postorder traversal \*/

#include <bits/stdc++.h>

using namespace std;

/\* A binary tree node has data,

pointer to left child and a

pointer to right child \*/

struct node

{

int data;

struct node \*left, \*right;

};

// A utility function to create a node

struct node\* newNode (int data)

{

struct node\* temp =

(struct node \*) malloc(sizeof(struct node));

temp->data = data;

temp->left = temp->right = NULL;

return temp;

}

// A recursive function to construct

// BST from post[]. postIndex is used

// to keep track of index in post[].

struct node\* constructTreeUtil(int post[], int\* postIndex,

int key, int min, int max,

int size)

{

// Base case

if (\*postIndex < 0)

return NULL;

struct node\* root = NULL;

// If current element of post[] is

// in range, then only it is part

// of current subtree

if (key > min && key < max)

{

// Allocate memory for root of this

// subtree and decrement \*postIndex

root = newNode(key);

\*postIndex = \*postIndex - 1;

if (\*postIndex >= 0)

{

// All nodes which are in range {key..max}

// will go in right subtree, and first such

// node will be root of right subtree.

root->right = constructTreeUtil(post, postIndex,

post[\*postIndex],

key, max, size );

// Construct the subtree under root

// All nodes which are in range {min .. key}

// will go in left subtree, and first such

// node will be root of left subtree.

root->left = constructTreeUtil(post, postIndex,

post[\*postIndex],

min, key, size );

}

}

return root;

}

// The main function to construct BST

// from given postorder traversal.

// This function mainly uses constructTreeUtil()

struct node \*constructTree (int post[],

int size)

{

int postIndex = size-1;

return constructTreeUtil(post, &postIndex,

post[postIndex],

INT\_MIN, INT\_MAX, size);

}

// A utility function to print

// inorder traversal of a Binary Tree

void printInorder (struct node\* node)

{

if (node == NULL)

return;

printInorder(node->left);

cout << node->data << " ";

printInorder(node->right);

}

// Driver Code

int main ()

{

int post[] = {15, 2, 25, 45, 35, 23, 100, 5 };

int size = sizeof(post) / sizeof(post[0]);

struct node \*root = constructTree(post, size);

cout << "Inorder traversal of "

<< "the constructed tree: \n";

printInorder(root);

return 0;

}

Output :-

