ASSIGNMENT 1

**Q1.** Develop a Menu driven program to demonstrate the following operations of

Arrays

——MENU——-

1.CREATE

2.DISPLAY

3.INSERT

4.DELETE

5.SEARCH

6.EXIT

**Ans:**

#include<iostream>

using namespace std;

int main()

{

int c = 6, n = 0, max = 20;

int \* arr;

do

{

cout<<"--MENU--\n1.CREATE\n2.DISPLAY\n3.INSERT\n4.DELETE\n5.SEARCH\n6.EXIT\n";

cin>>c;

switch(c)

{

case 1:

{

cout<<"Enter size of array: ";

cin>>n;

cout<<"Enter array: ";

arr = new int[n];

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

{

cin>>arr[i];

}

cout<<"\n";

break;

}

case 2:

{

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

{

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

}

cout<<"\n\n";

break;

}

case 3:

{

int index, e, temp, x = 0;

cout<<"Enter index to insert element at: ";

cin>>index;

cout<<"Enter element to add: ";

cin>>e;

if(n == max)

{

cout<<"Error: max size reached\n\n";

break;

}

temp = arr[index - 1];

arr[index - 1] = e;

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

{

temp += arr[i];

arr[i] = temp - arr[i];

temp = temp - arr[i];

}

arr[n] = temp;

n++;

cout<<"\n\n";

break;

}

case 4:

{

int index, temp;

cout<<"Enter index to delete: ";

cin>>index;

temp = arr[n-1];

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

{

temp += arr[i];

arr[i] = temp - arr[i];

temp = temp - arr[i];

}

arr[index - 1] = temp;

n--;

cout<<"\n\n";

break;

}

case 5:

{

int e;

cout<<"Enter element to search: ";

cin>>e;

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

{

if(arr[i] == e)

{

cout<<"Element "<<e<<" found at index "<<i + 1<<"\n";

}

}

cout<<"\n";

break;

}

default:

c = 6;

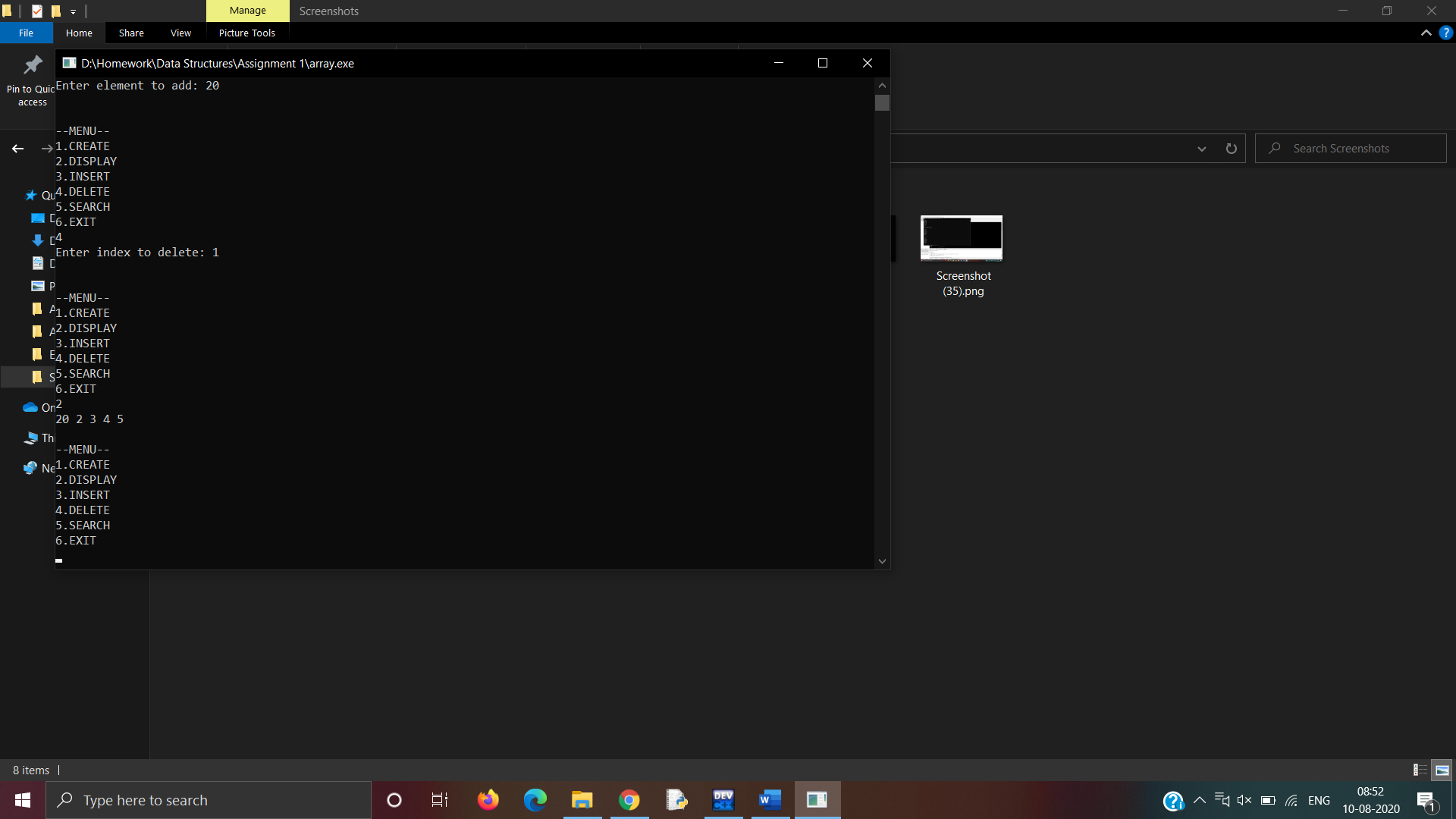
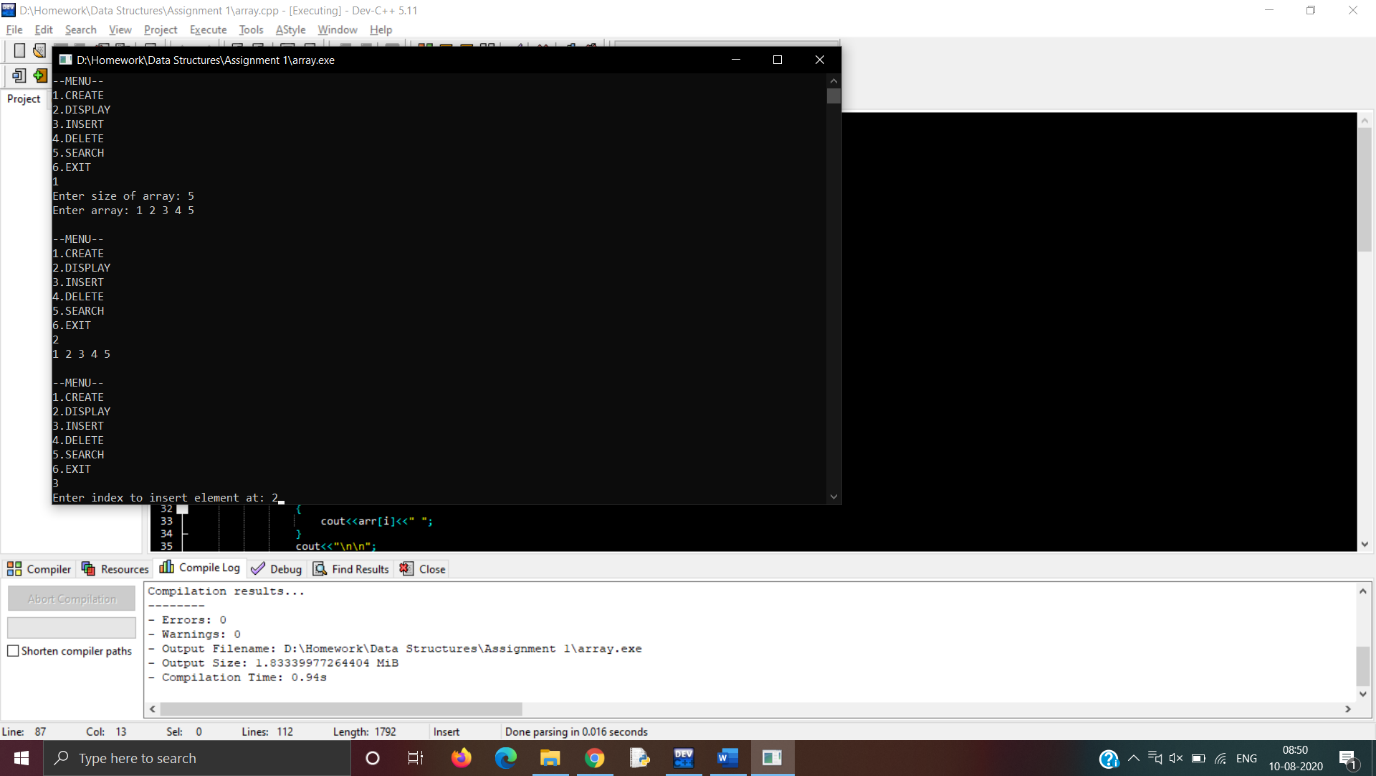
}

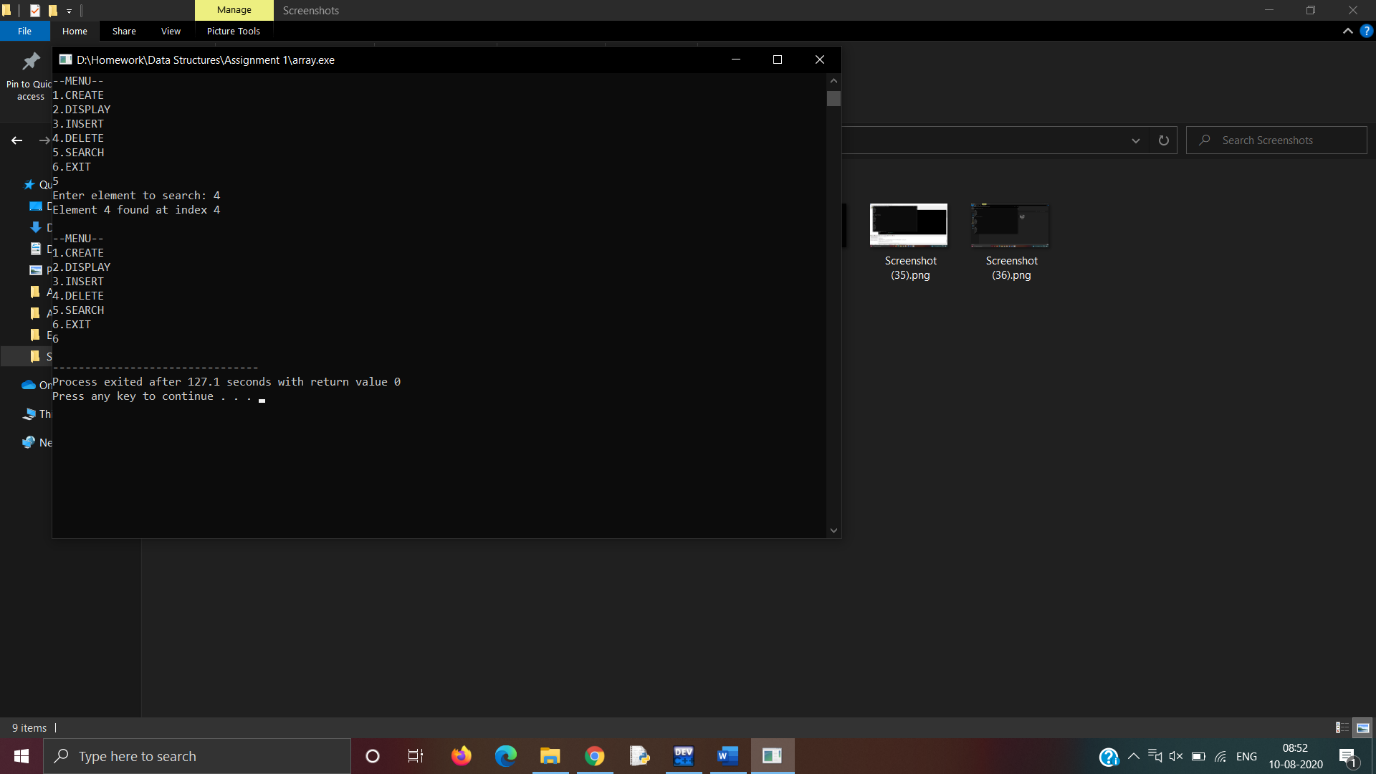
}while(c != 6);

return 0;

}

**Output:**

** **

****

**Q2. Design the logic to remove the duplicate elements from an Array and after the deletion the array should contain the unique elements.**

**Ans:**

#include <iostream>

using namespace std;

int main()

{

int x = 0, n, arr[100], temp;

cout<<"Enter size of array: ";

cin>>n;

cout<<"Enter an array: ";

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

{

int flag = 1;

cin>>temp;

for(int j = 0; j < x; j++)

{

if(temp == arr[j])

{

flag = 0;

}

}

if(flag)

{

arr[x] = temp;

x++;

}

}

cout<<"The changed array is: ";

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

{

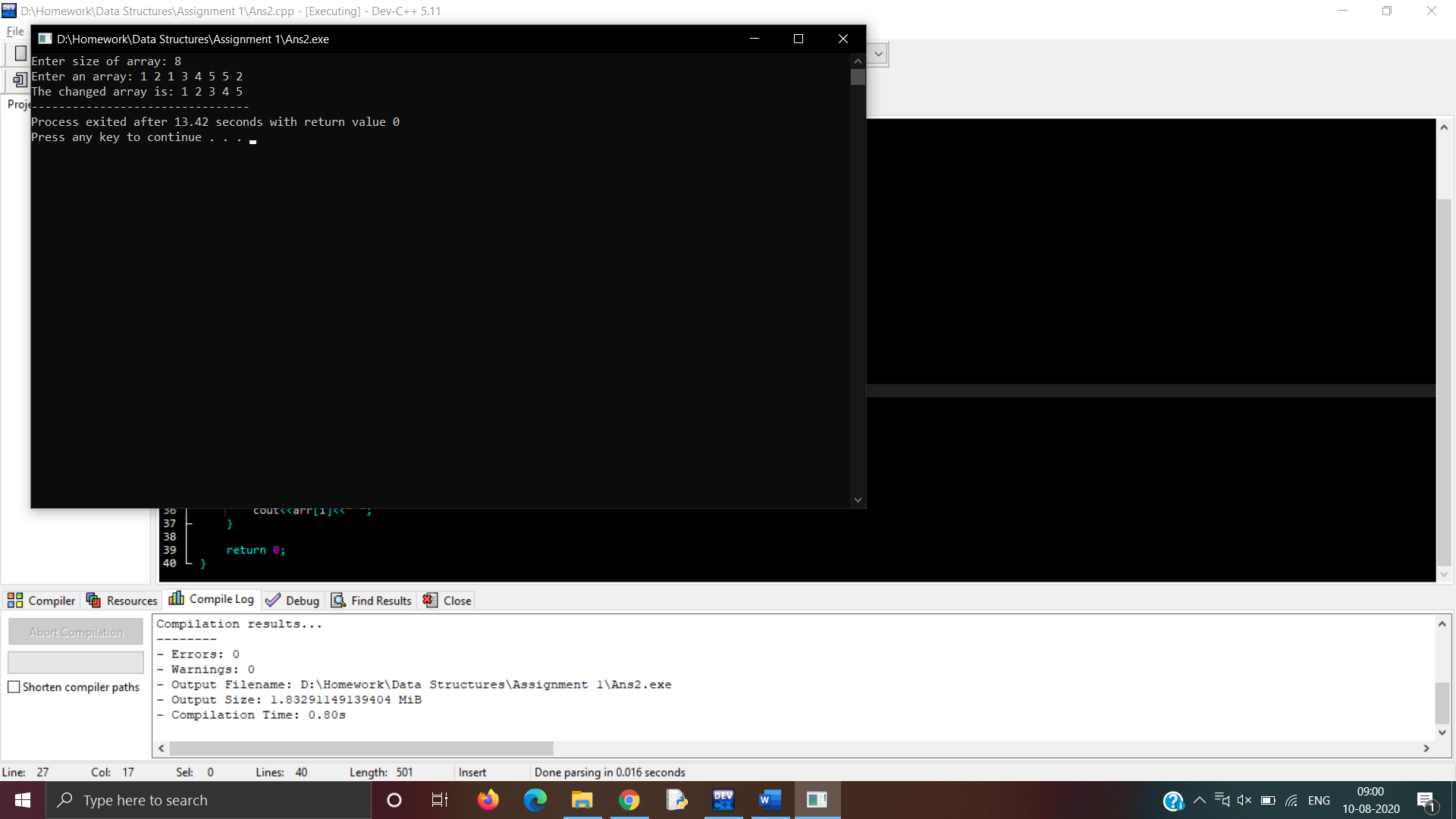
cout<<arr[i]<<" ";

}

return 0;

}

**Output:**

****

**Q3. Predict the Output of the following program**

**int main()**

**{**

**int i;**

**int arr[5] = {1};**

**for (i = 0; i < 5; i++)**

**printf("%d ", arr[i]);**

**return 0;**

**}**

**Ans:**

1 0 0 0 0

**Q4. Implement the logic to**

**i) Reverse the elements of an array**

**Ans:**

#include<iostream>

using namespace std;

int main()

{

int n, \*arr, temp;

cout<<"Enter number of elements in the array: ";

cin>>n;

arr = new int[n];

cout<<"Enter the array: ";

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

{

cin>>arr[i];

}

for(int i = 0; i < n/2; i++)

{

temp = arr[i];

arr[i] = arr[n - i - 1];

arr[n - i - 1] = temp;

}

cout<<"The reversed array is: ";

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

{

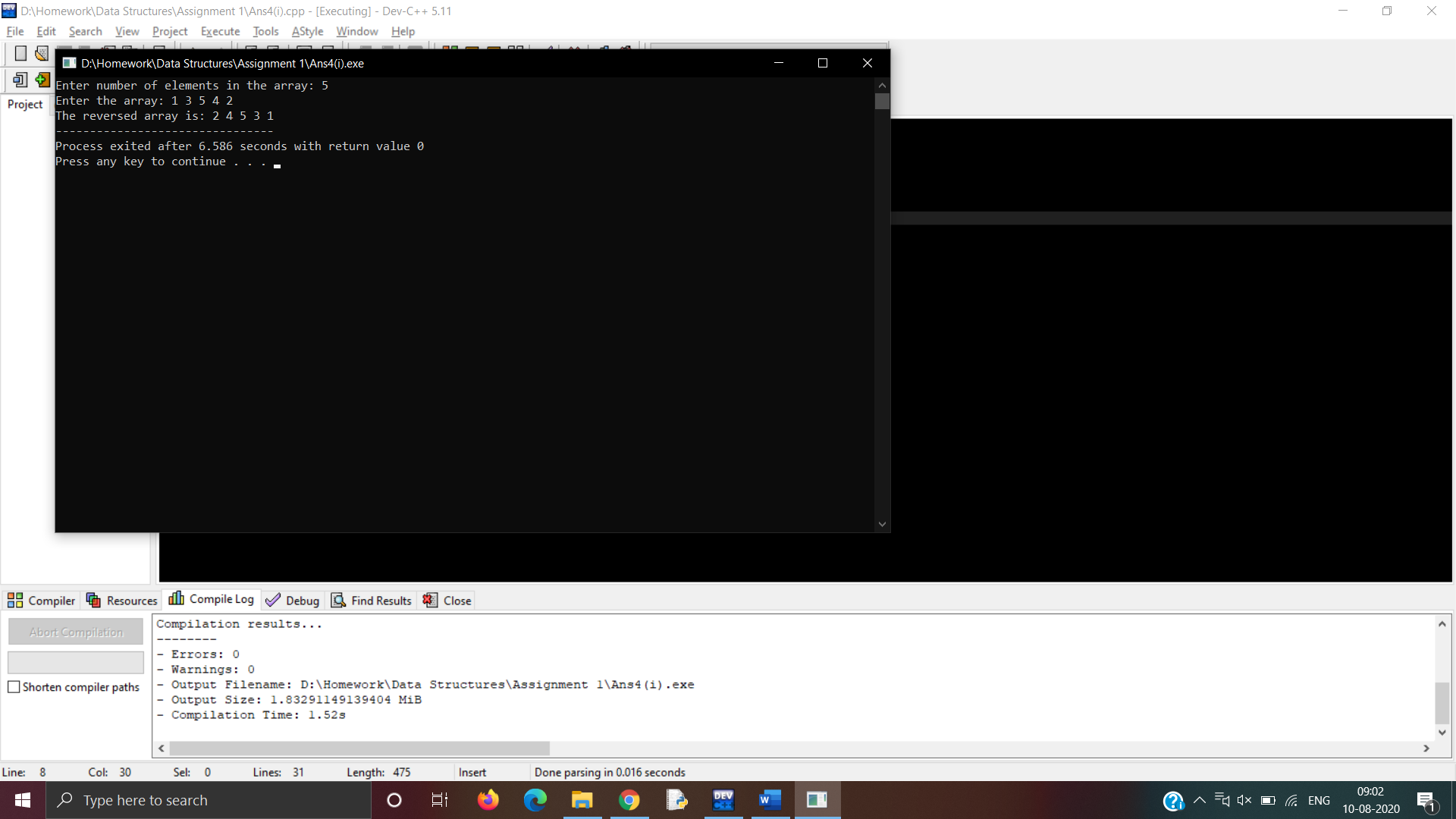
cout<<arr[i]<<" ";

}

return 0;

}

**Output:**

****

**ii) Find the matrix multiplication**

**Ans:**

#include<iostream>

using namespace std;

int main()

{

int a[3][3], b[3][3], c[3][3];

cout<<"Enter matrix A: ";

for(int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

cin>>a[i][j];

}

}

cout<<"Enter matrix B: ";

for(int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

cin>>b[i][j];

}

}

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

c[i][j] = 0;

for(int k = 0; k < 3; k++)

{

c[i][j] += a[i][k] \* b[k][j];

}

}

}

cout<<"The multiplication of the above matrices is:\n";

for(int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

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

}

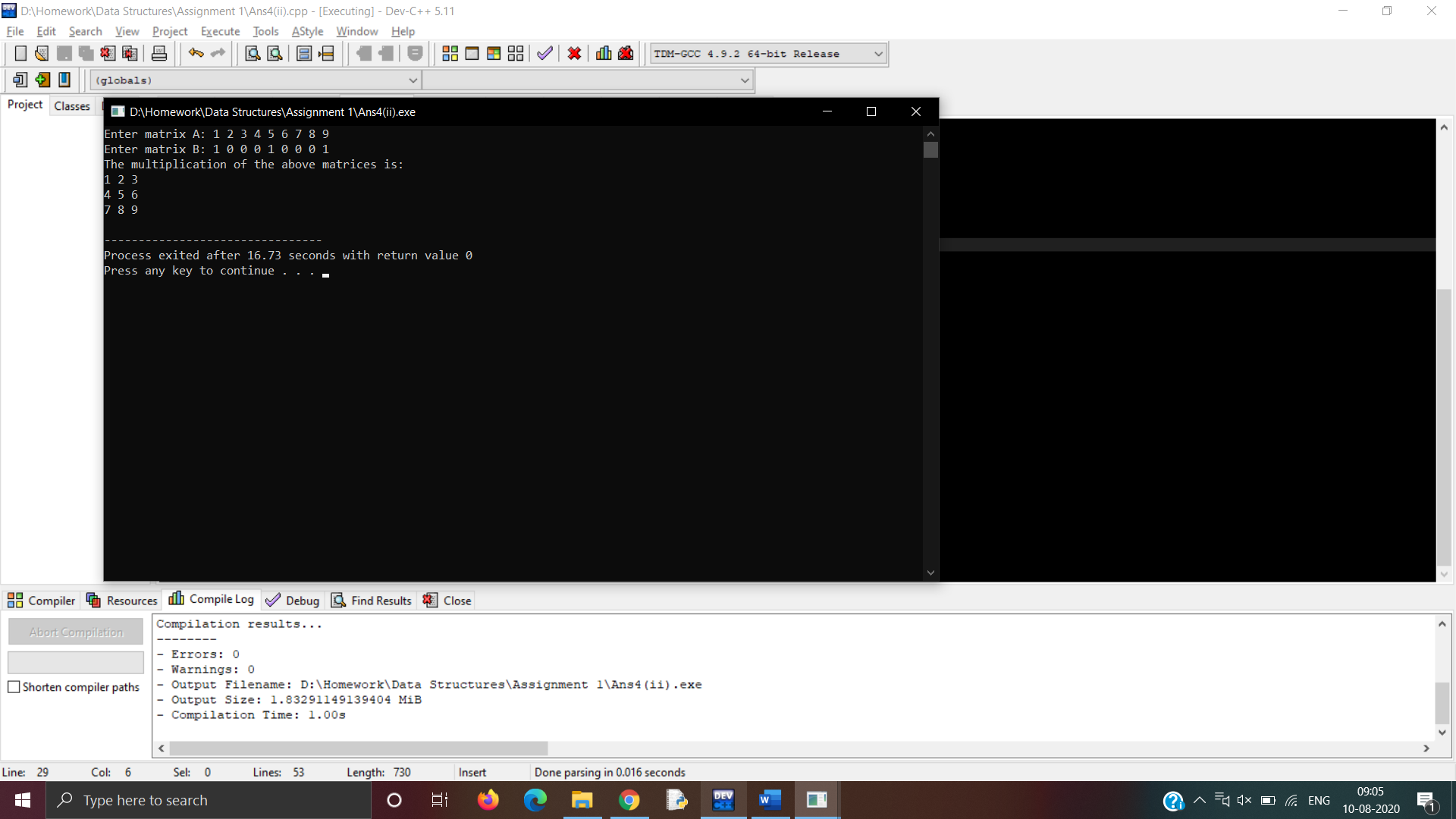
cout<<"\n";

}

return 0;

}

**Output:**

****

**iii) Find the Transpose of a Matrix**

**Ans:**

#include<iostream>

using namespace std;

int main ()

{

int a[3][3], b[3][3];

cout<<"Enter matrix: ";

for(int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

cin>>a[i][j];

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

}

}

cout<<"The transpose is:\n";

for(int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

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

}

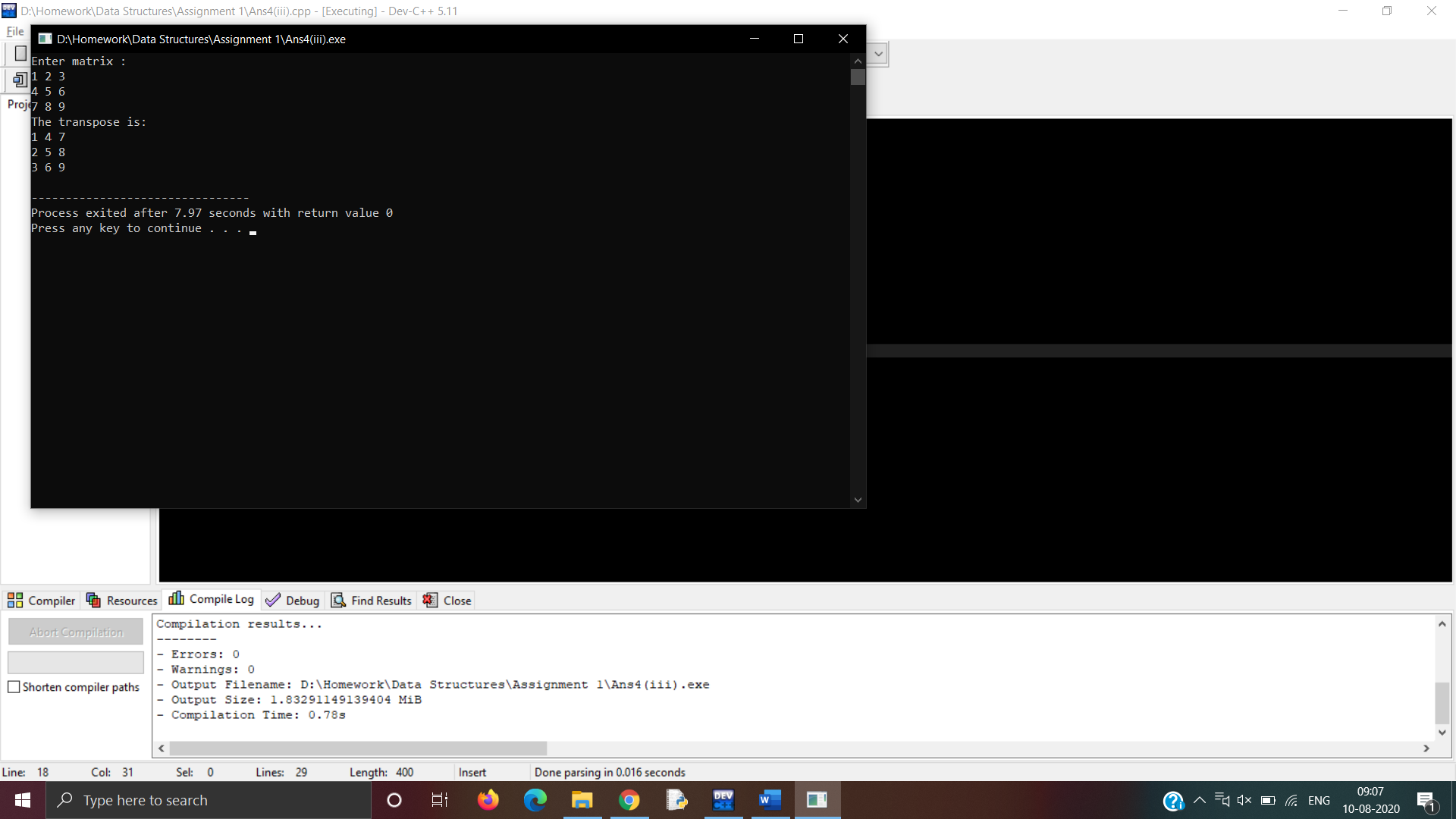
cout<<"\n";

}

return 0;

}

**Output:**

****

**Q5. Implement the Binary search algorithm regarded as a fast search**

**algorithm with run-time complexity of Ο(log n) in comparison to the Linear**

**Search.**

**Ans:**

#include<iostream>

using namespace std;

int search(int size, int \*arr, int e)

{

int head = 0, tail = size - 1, mid;

while (head < tail)

{

mid = (head + tail)/2;

if(arr[mid] == e)

{

return mid;

}

if(arr[mid] < e)

{

tail = mid - 1;

}

else

{

head = mid + 1;

}

}

return -1;

}

int main()

{

int n, \*arr, e;

cout<<"Enter size of array: ";

cin>>n;

arr = new int[n];

cout<<"Enter sorted array: ";

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

{

cin>>arr[i];

}

cout<<"Enter element to search: ";

cin>>e;

int x = search(n, arr, e);

if(x == -1)

{

cout<<"Error: Element not found";

}

else

{

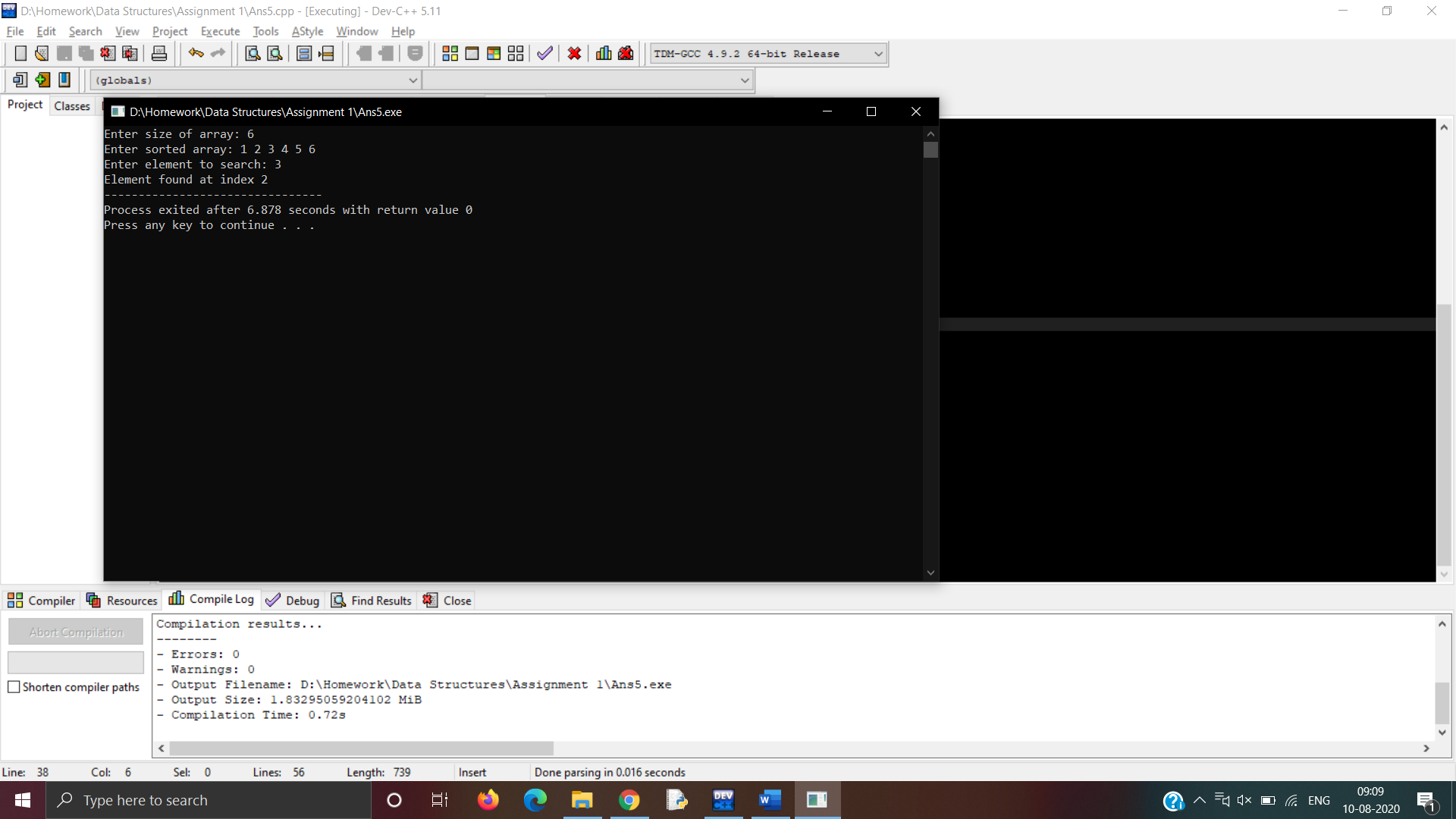
cout<<"Element found at index "<<x;

}

return 0;

}

**Output:**

****

**Q6. Bubble Sort is the simplest sorting algorithm that works by repeatedly**

**swapping the adjacent elements if they are in wrong order. Code the**

**Bubble sort with the following elements: 64 34 25 12 22 11 90.**

**Ans:**

#include<iostream>

using namespace std;

int main()

{

int n = 7, arr[7] = {64, 34, 25, 12, 22, 11, 90};

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

{

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

{

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

{

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

cout<<"The sorted array is: ";

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

{

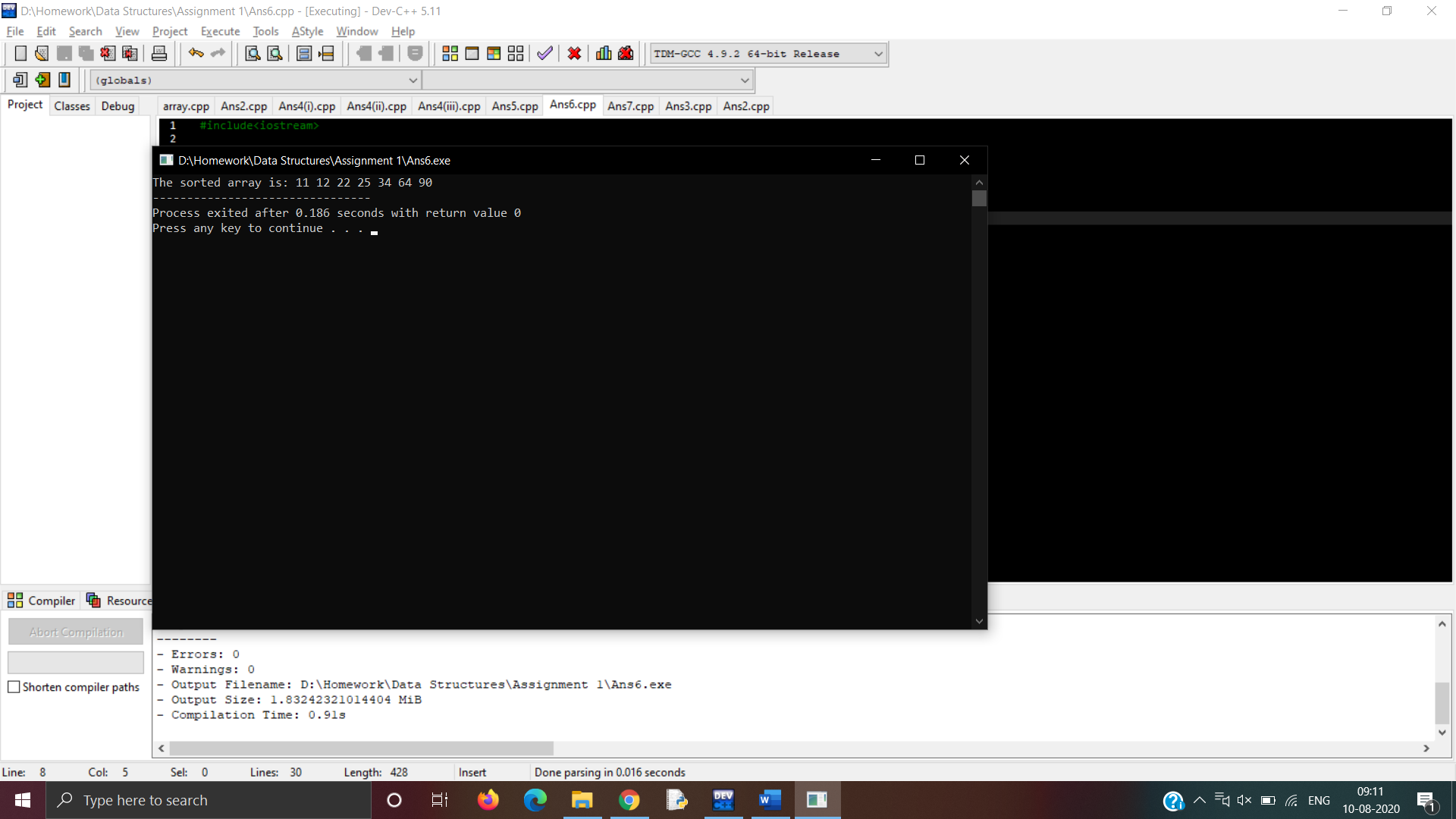
cout<<arr[i]<<" ";

}

return 0;

}

**Output:**

****

**Q7. Design the Logic to Find a Missing Number in a Sorted Array.**

**Ans:**

#include <iostream>

using namespace std;

int main()

{

int n, \*arr, diff = -1, temp, index;

cout<<"Enter array length: ";

cin>>n;

arr = new int[n];

cout<<"Enter an array: ";

cin>>arr[0];

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

{

cin>>arr[i];

if(diff == -1)

{

diff = arr[i] - arr[i - 1];

}

else

{

if(diff < arr[i] - arr[i - 1])

{

index = i;

}

else if(diff > arr[i] - arr[i - 1])

{

index = i - 1;

}

}

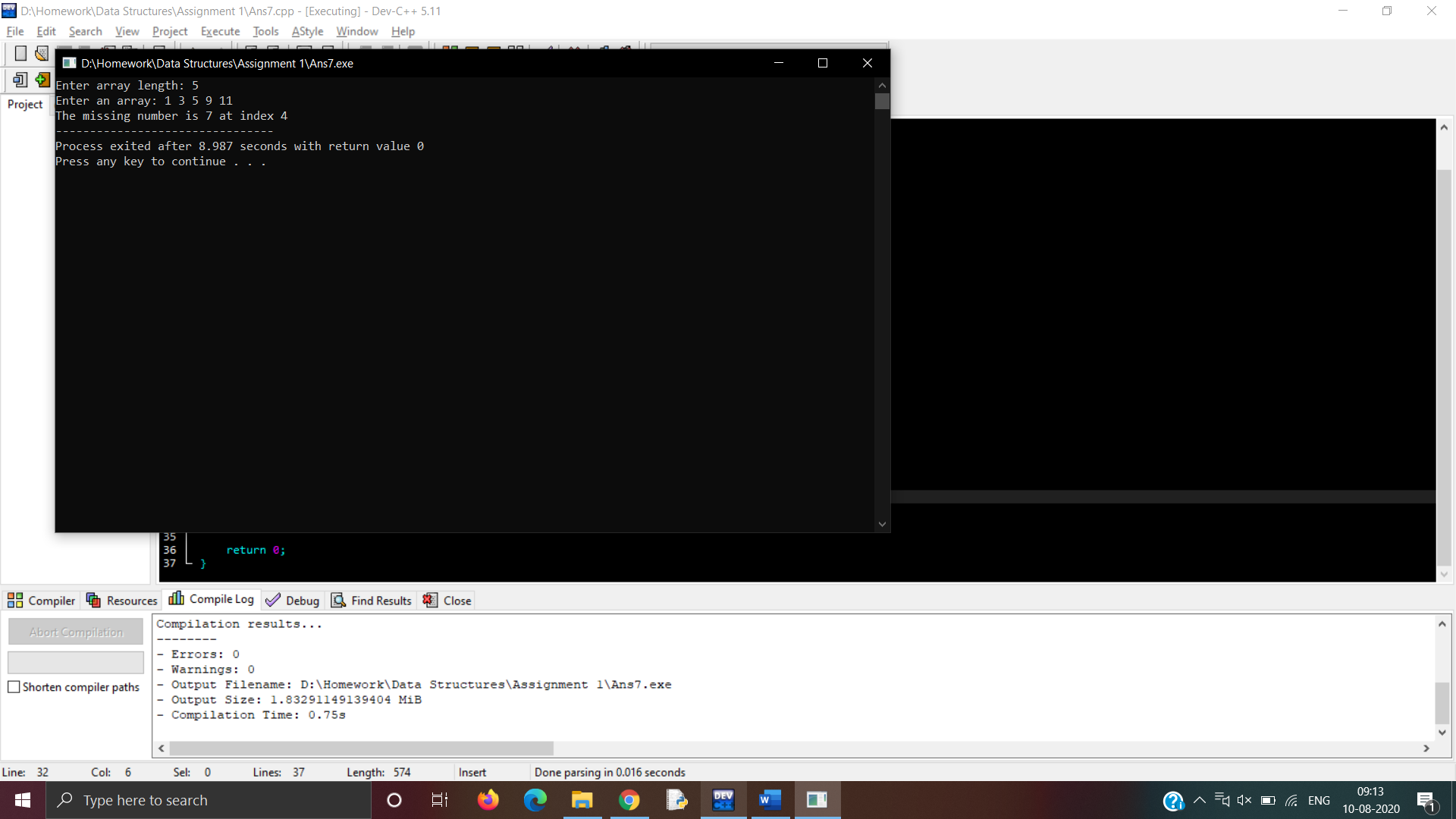
}

cout<<"The missing number is "<<arr[index] - diff<<" at index "<<index + 1;

return 0;

}

**Output:**

****