**DAA LAB**

**1.** **Given an array of nonnegative integers, design a linear algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(n), where n is the size of input)**

**1.Linear Search**

**CODE:**

#include<iostream>

#define max

using namespace std;

void linear(int a[],int n,int key)

{

int i,j,comp=0;

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

{

if(a[i]=key)

{

++comp;

j==1;

//cout<<"Present"<<endl;

break;

}

else

++comp;

j=0;

//cout<<"Not Present"<<endl;

}

cout<<"Total comparisions"<<comp<<endl;

if(j==1)

cout<<"present"<<endl;

else

cout<<"Not present"<<endl;

}

int main()

{

int n,key,a[n],t,i;

cout<<"Enter the no of test cases"<<endl;

cin>>t;

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

{

cout<<"Enter the Size of array: "<<endl;

cin>>n;

cout<<"Enter the elements of array"<<endl;

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

{

cin>>a[i];

}

cout<<"Enter the element to be searched:"<<endl;

cin>>key;

linear(a,n,key);

}

}

**OUTPUT:**

enter the no of test cases

3

enter the size of an array

8

34 35 65 31 25 89 64 30

enter the element to be searched

89

present

Total Comparisons: 6

enter the size of an array

5

977 354 244 546 355

enter the element to be searched

244

present

Total Comparisons: 3

enter the size of an array

6

23 64 13 67 43 56

enter the element to be searched

63

Total Comparisons:6

not present

Process returned 0 (0x0) execution time : 83.227 s

Press any key to continue.

**2.Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(nlogn), where n is the size of input).**

**2.Binary SearchBinary Search**

**CODE:**

#include<iostream>

using namespace std;

int binarysearch(int a[],int i,int j,int key)

{

while(i<j)

{

int mid,comp=0;

mid=i+(j-1)/2;

if(a[mid]==key)

{

comp++;

return mid;

}

if(a[mid]>key)

{

comp++;

return mid-1;

}

else

{

comp++;

return mid+1;

}

cout<<"Total Comparisons:"<<comp<<endl;

}

return -1;

}

int main()

{

int n,key,t,i;

cout<<"Enter the no of test cases"<<endl;

cin>>t;

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

{

cout<<"Enter the Size of array: "<<endl;

cin>>n;

cout<<"Enter the elements of array"<<endl;

int a[n];

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

{

cin>>a[i];

}

cout<<"Enter the element to be searched:"<<endl;

cin>>key;

int res=binarysearch(a,0,n-1,key);

if(res==-1)

cout<<"Not Present"<<endl;

else

cout<<"Present"<<endl;

}

}

OUTPUT:

Enter the no of test cases

3

Enter the Size of array:

5

Enter the elements of array

12 23 36 39 41

Enter the element to be searched:

41

Present

Enter the Size of array:

8

Enter the elements of array

21 39 40 45 51 54 68 72

Enter the element to be searched:

69

Not Present

Enter the Size of array:

10

Enter the elements of array

101 246 438 561 796 896 899 4644 7999 8545

Enter the element to be searched:

7999

Present

Process returned 0 (0x0) execution time : 131.561 s

Press any key to continue.

**3.Jump Search3. Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether a given key element is present in the sorted array or not. For an array arr[n], search at the indexes arr[0], arr[2], arr[4],.....,arr[2k ] and so on. Once the interval (arr[2k ] < key < arr[ 2k+1] ) is found, perform a linear search operation from the index 2k to find the element key. (Complexity < O(n), where n is the number of elements need to be scanned for searching):**

**3.Jump SearchJump Search**

CODE:

#include<iostream>

#include<math.h>

using namespace std;

void jumpsearch(int a[],int n,int key)

{

int start=0,comp;

int end=sqrt(n);

while(a[end]<=key&&end<n)

{

comp++;

start=end;

end+=sqrt(n);

if(end>n-1)

end=n;

}

for(int i=start;i<end;i++)

{

if(a[i]==key)

{

comp++;

cout<<"Present "<<endl;

}

else

cout<<"Not Present"<<endl;

}

}

int main()

{

int n,key,t,i;

cout<<"Enter the no of test cases"<<endl;

cin>>t;

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

{

cout<<"Enter the Size of array: "<<endl;

cin>>n;

cout<<"Enter the elements of array"<<endl;

int a[n];

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

{

cin>>a[i];

}

cout<<"Enter the element to be searched:"<<endl;

cin>>key;

jumpsearch(a,n,key);

}

}

**Output:**

Enter the no of test cases

3

Enter the Size of array:

5

Enter the elements of array

12 23 36 39 41

Enter the element to be searched:

41

Present

Enter the Size of array:

8

Enter the elements of array

21 39 40 45 51 54 68 72

Enter the element to be searched:

69

Not Present

Enter the Size of array:

10

Enter the elements of array

101 246 438 561 796 896 899 4644 7999 8545

Enter the element to be searched:

7999

Present

**4.Given a sorted array of positive integers containing few duplicate elements, design an algorithm and implement it using a program to find whether the given key element is present in the array or not. If present, then also find the number of copies of given key. (Time Complexity = O(log n))**

**CODE:**

#include <iostream>

#define max

using namespace std;

int main()

{

int n,t,key;

cout<<"Enter the number of test cases:" <<endl;

cin>>t;

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

{

cout<<"Enter the size of Araay:"<<endl;

cin>>n;

cout<<"Enter the elements of array:"<<endl;

int a[n],count=0;

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

{ cin>>a[j]; }

cout<<"Enter the element to be found"<<endl;

cin>>key;

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

{

if(a[j]=key)

count++;

}

if(count==0)

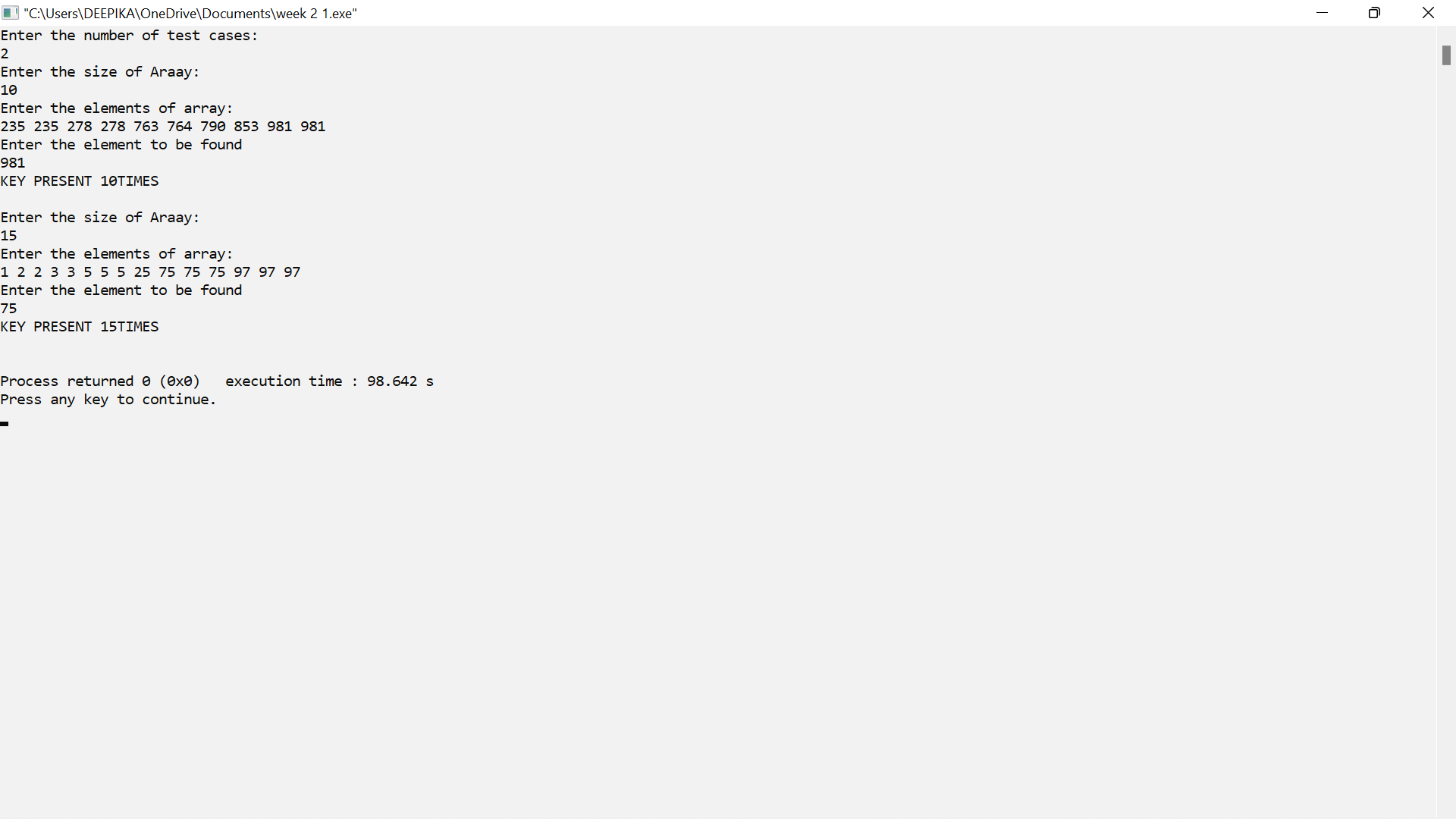
cout<<"KEY NOT PRESENT"<<endl;

else

cout<<"KEY PRESENT"<<" "<<count<<"TIMES"<<endl;

cout<<endl;} }

**OUTPUT:**

****



3



**5.** **Given a sorted array of positive integers, design an algorithm and implement it using a program to find three indices i, j, k such that arr[i] + arr[j] = arr[k].**

**CODE:**

#include <iostream>

#define max

using namespace std;

int main()

{

int n,t,key;

cout<<"Enter the number of test cases:" <<endl;

cin>>t;

for(int z=0;z<t;z++)

{

cout<<"Enter the size of Araay:"<<endl;

cin>>n;

cout<<"Enter the elements of array:"<<endl;

int a[n];

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

{

cin>>a[i];

}

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

{

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

{

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

{

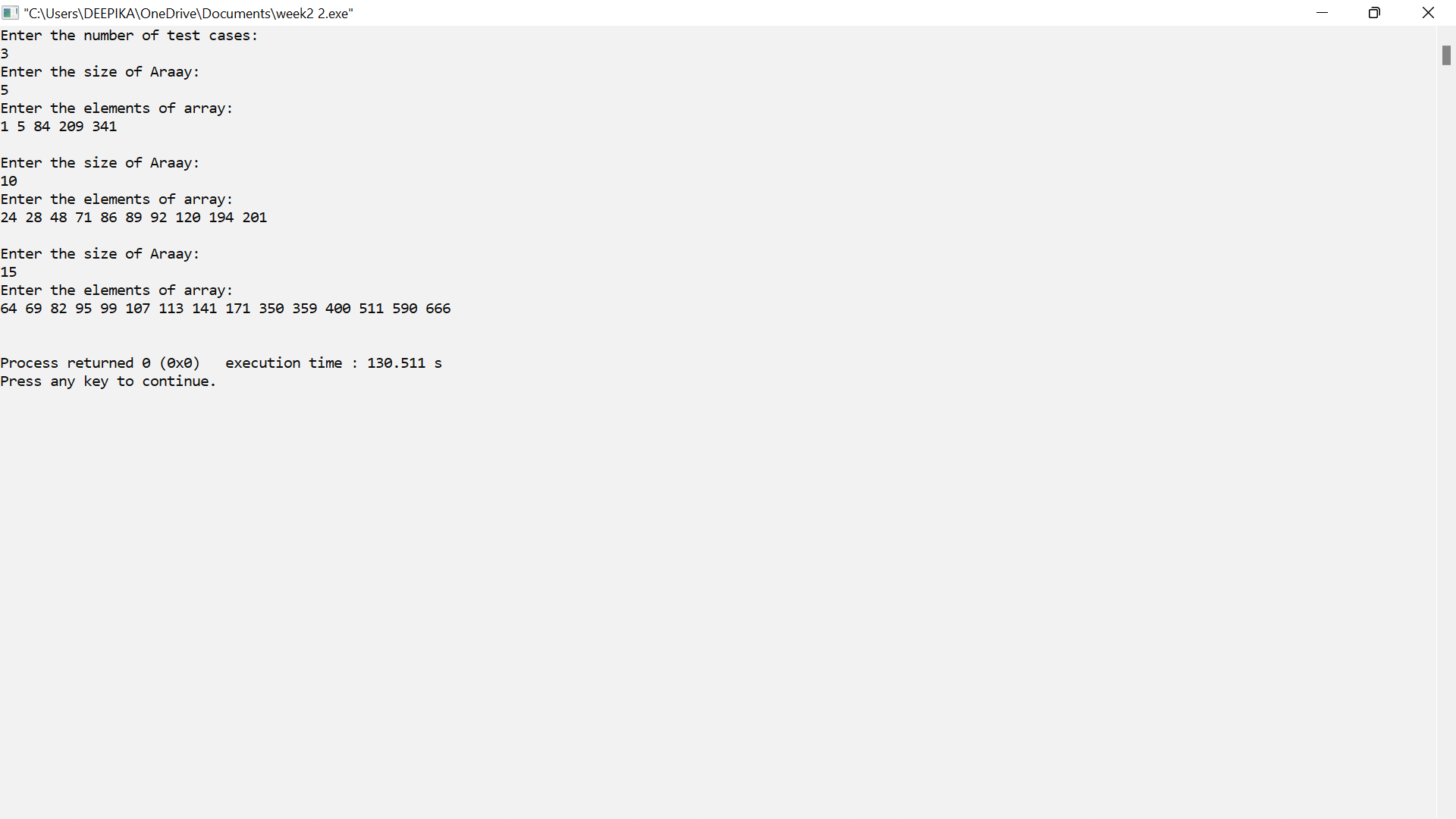
if(a[i]+a[j]==a[k])

cout<<i+1<<""<<j+1<<""<<k+1<<endl;

}

} } }

**OUTPUT:**

****

**6.** **Given an array of nonnegative integers, design an algorithm and a program to count the number of pairs of integers such that their difference is equal to a given key, K.**

**CODE:**

#include <iostream>

#define max

using namespace std;

int main()

{

int n,t,key;

cout<<"Enter the number of test cases:" <<endl;

cin>>t;

while(t--)

{

cout<<"Enter the size of Araay:"<<endl;

cin>>n;

cout<<"Enter the elements of array:"<<endl;

int a[n],c=0;

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

{

cin>>a[i];

}

cout<<"Enter the element to check difference"<<endl;

cin>>key;

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

{

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

{

if(a[i]-a[j]==key||a[j]-a[i]==key)

{

c++;

}

}

}

if(c==0)

cout<<"Difference not present"<<endl;

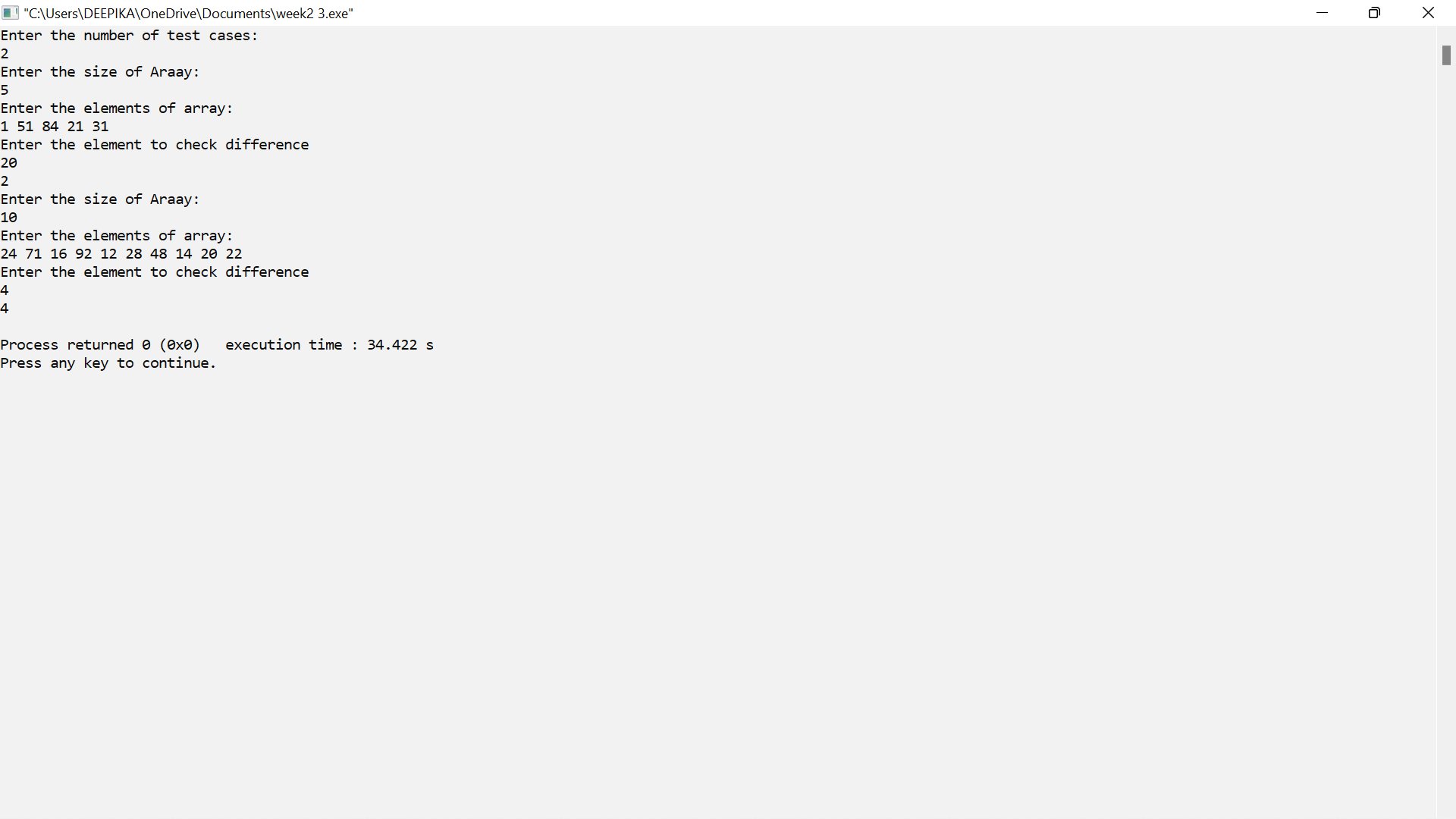
else

cout<<c<<endl;

}

}

**OUTPUT:**

****

**7. Given an unsorted array of integers, design an algorithm and a program to sort the array using insertion sort. Your program should be able to find number of comparisons and shifts ( shifts - total number of times the array elements are shifted from their place) required for sorting the array.**

**7.INSERTION SORT.INSERTION SORT**

CODE:

#include <iostream>

using namespace std;

void insertion(int a[],int n)

{

int min,j,i;

int comp=0,shifts=0;

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

{

min=a[i];

j=i-1;

while(j>=0&&min<a[j])

{

shifts++;

comp++;

a[j+1]=a[j];

j--;

}

shifts++;

a[j+1]=min;

}

cout<<"Total comparisions:"<<comp<<endl;

cout<<"Total shifts:"<<shifts<<endl;

}

int main()

{

int n,t;

cout<<"Enter the number of test cases:" <<endl;

cin>>t;

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

{

cout<<"Enter the size of Araay:"<<endl;

cin>>n;

cout<<"Enter the elements of array:"<<endl;

int a[n];

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

{

cin>>a[j];

}

insertion(a,n);

cout<<"Sorted array:"<<endl;

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

{

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

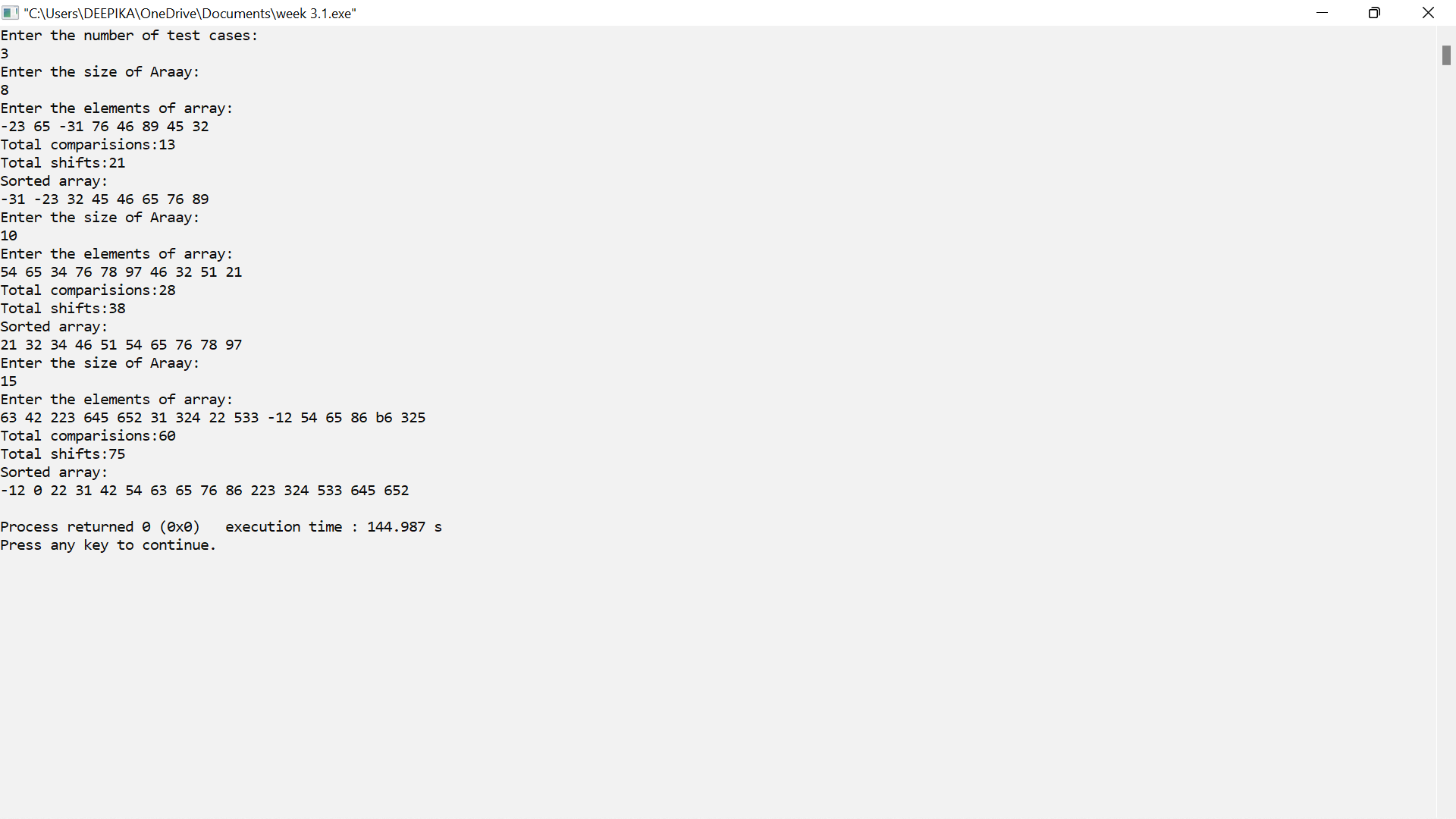
}

cout<<endl;

}

}

**OUTPUT:**



**8. Given an unsorted array of integers, design an algorithm and implement a program to sort this array using selection sort. Your program should also find number of comparisons and number of swaps required.**

**8.SELECTION SORTSELECTION SORT**

**CODE:**

#include<iostream>

using namespace std;

void selection(int a[],int n)

{

int i,j,min=0,pos;

int swaps,comp;

for(i=0;i<n-1;i++)

{

min=a[i];

pos=i;

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

{

if(min>a[j])

{

min=a[j];

pos=j;

}

comp++;

}

if(pos!=i)

{

a[pos]=a[i];

a[i]=min;

swaps++;

}

}

cout<<"Total swaps:"<<swaps<<endl;

cout<<"Total comparisions:"<<comp<<endl;

}

int main()

{

int n,t;

cout<<"Enter the number of test cases:" <<endl;

cin>>t;

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

{

cout<<"Enter the size of Araay:"<<endl;

cin>>n;

cout<<"Enter the elements of array:"<<endl;

int a[n];

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

{

cin>>a[j];

}

selection(a,n);

cout<<"Sorted array:"<<endl;

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

{

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

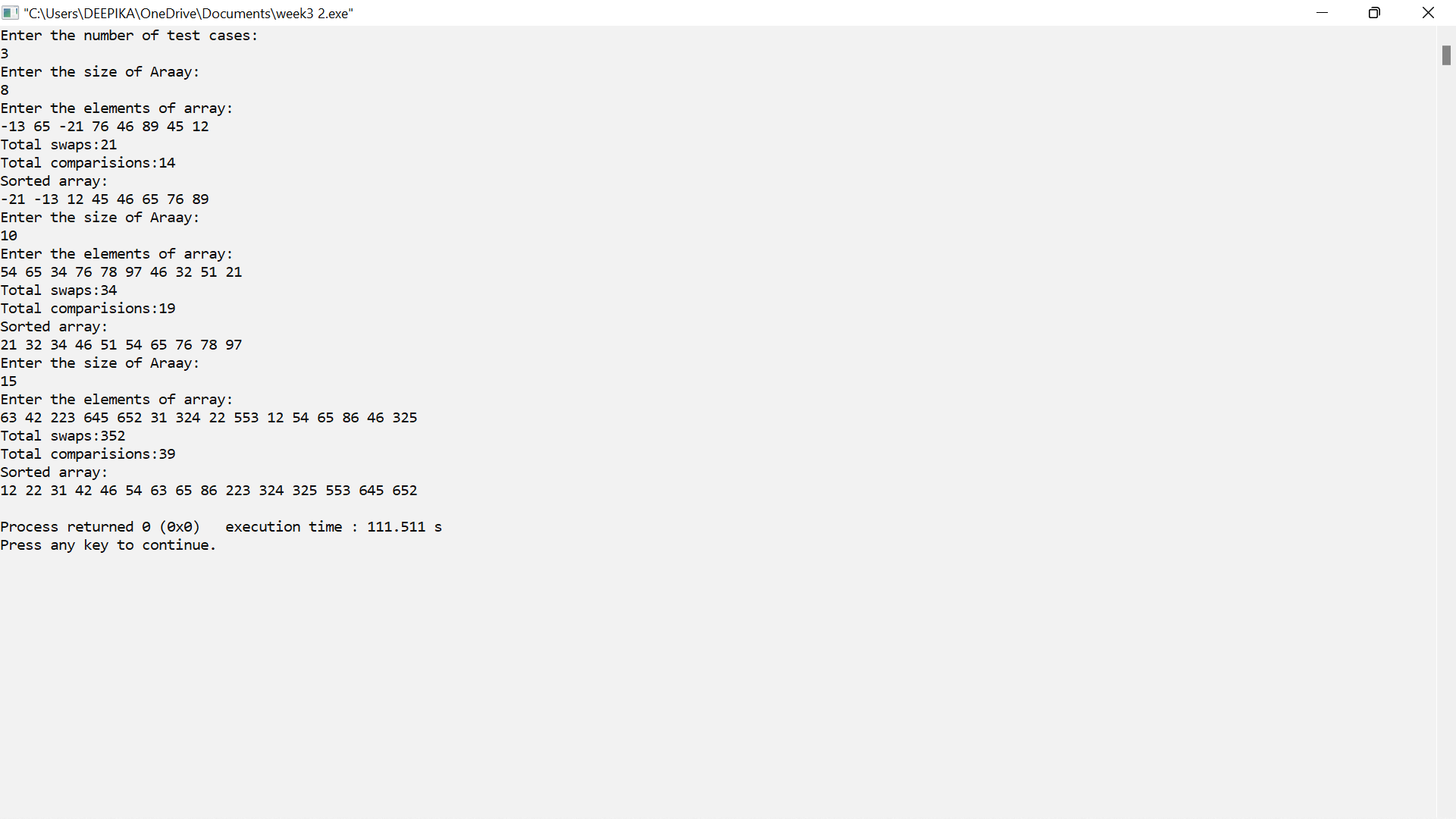
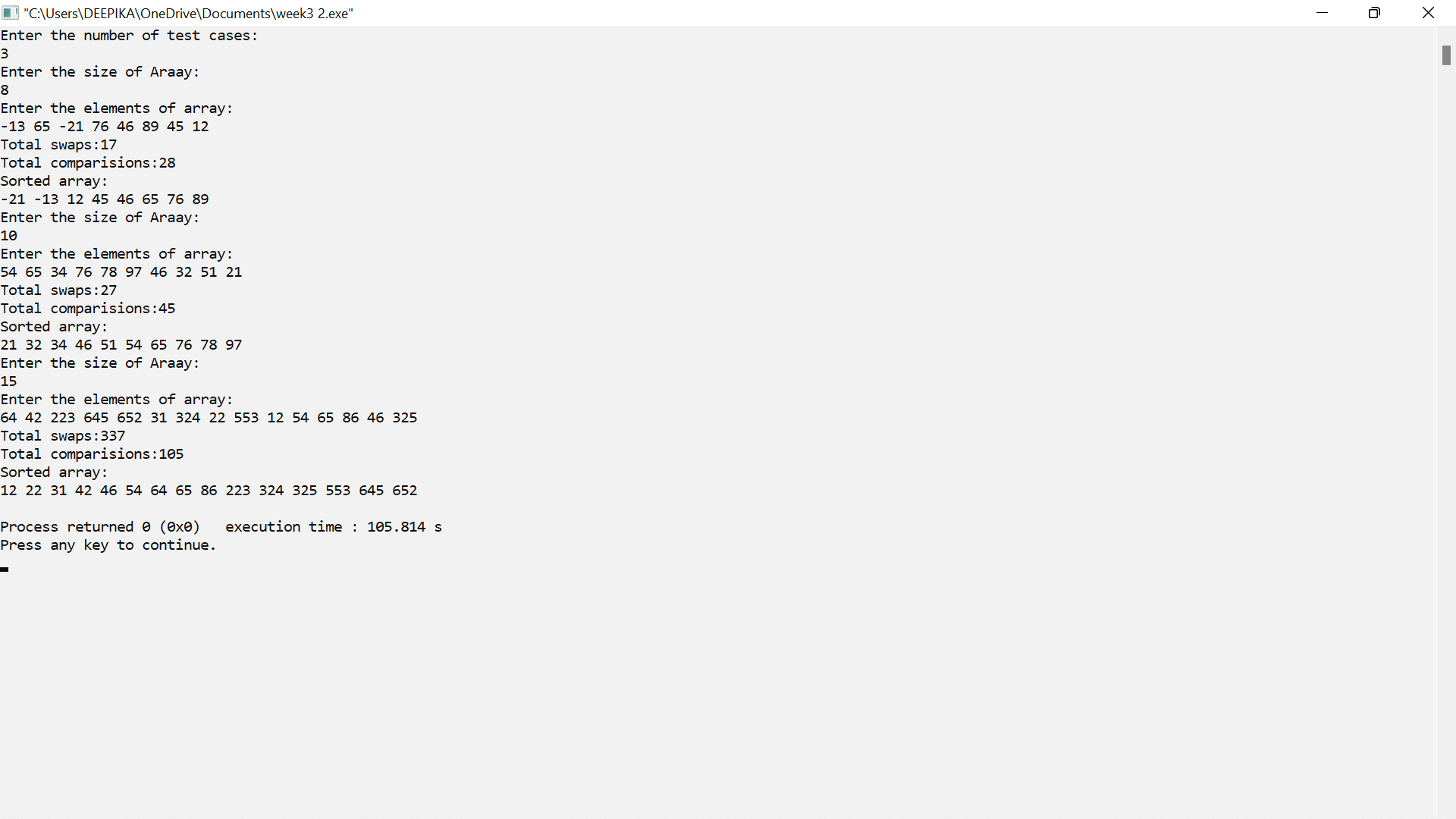
}

cout<<endl;

}

}

**OUTPUT:**



**9.** **Given an unsorted array of positive integers, design an algorithm and implement it using a program to find whether there are any duplicate elements in the array or not. (use sorting) (Time Complexity = O(n log n))**

**TO CHECK WEATHER DUPLICATE ELEMENT IS PRESENT OR NOT**

**CODE:**

#include <bits/stdc++.h>

using namespace std;

void findDuplicates(int arr[], int len)

{

bool ifPresent = false;

// ArrayList to store the output

vector<int> al;

for(int i = 0; i < len - 1; i++)

{

for(int j = i + 1; j < len; j++)

{

if (arr[i] == arr[j])

{

auto it = std::find(al.begin(),al.end(), arr[i]);

if (it != al.end())

{

break;

}

else

{

al.push\_back(arr[i]);

ifPresent = true;

} } }

}

if (ifPresent == true)

{

cout << "YES"<<endl;

}

else

{

cout << "No"<<endl;

}

}

int main()

{

int i,len,t;

cout<<"Enter the number of test cases:" <<endl;

cin>>t;

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

{

cout<<"Enter the Array length: "<<endl;;

cin>>len;

int arr[len];

cout<<"\n";

cout<<"Enter the Number for array :"<<endl;

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

{

cin>>arr[i];

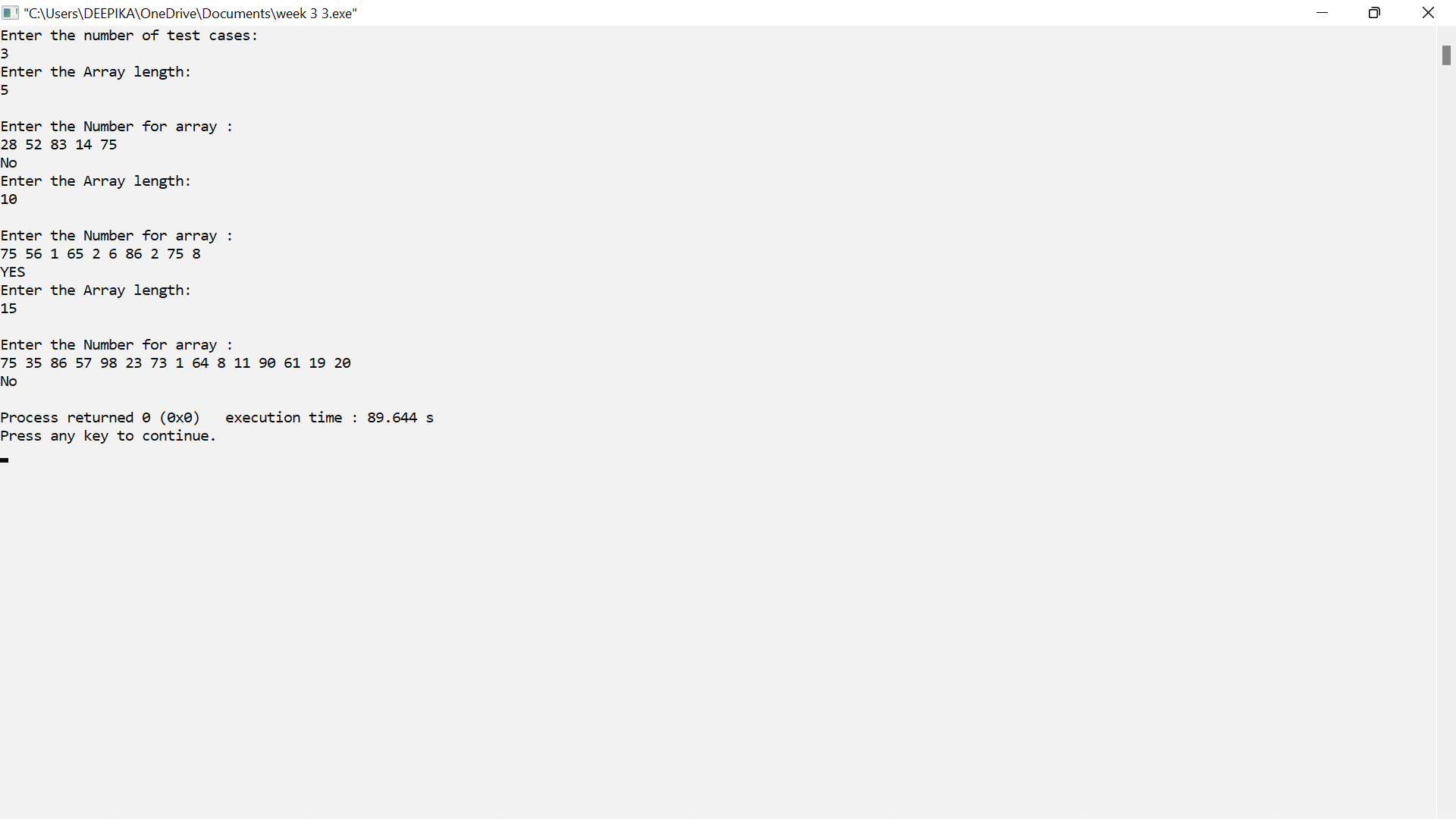
}

int n = sizeof(arr) / sizeof(arr[0]);

findDuplicates(arr, n);

} return 0; }

**OUTPUT:**



**10.** **Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by dividing the array into two subarrays and combining these subarrays after sorting each one of them. Your program should also find number of comparisons and inversions during sorting the array.**

**10.MERGE SORT**

**CODE:**

#include <bits/stdc++.h>

using namespace std;

int comp = 0,inv=0;

void mergeArray(int arr[], int l, int mid, int h)

{

int n1 = mid - l + 1;

int n2 = h - mid;

int a[n1], b[n2];

for(int i = 0; i < n1; i++)

{

a[i] = arr[l + i];

}

for(int i = 0; i < n2; i++)

{

b[i] = arr[mid + 1 + i];

}

int i = 0, j = 0, k = l;

while(i < n1 && j < n2)

{

comp++;

if(a[i] <= b[j])

{

arr[k] = a[i];

i++;

}

else

{

arr[k] = b[j];

j++;

inv++;

}

k++;

}

while(i < n1)

{

arr[k] = a[i];

i++; k++;

}

while(j < n2)

{

arr[k] = b[j];

j++; k++;

}

}

void mergeSort(int arr[], int l, int h)

{

if(l < h)

{

int mid = l + (h - l) / 2;

mergeSort(arr, l, mid);

mergeSort(arr, mid + 1, h);

mergeArray(arr, l, mid, h);

}

}

void display(int arr[], int n)

{

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

{

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

}

cout<<endl;

}

int main()

{

int t;

cout<<"Enter the number of test cases:" <<endl;

cin>>t;

while(t--)

{

cout<<"Enter the size of Araay:"<<endl;

int n;

cin>>n;

int arr[n];

cout<<"Enter the elements of array:"<<endl;

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

{

cin>>arr[i];

}

mergeSort(arr, 0, n - 1);

cout<<"Sorted array:"<<endl;

display(arr, n);

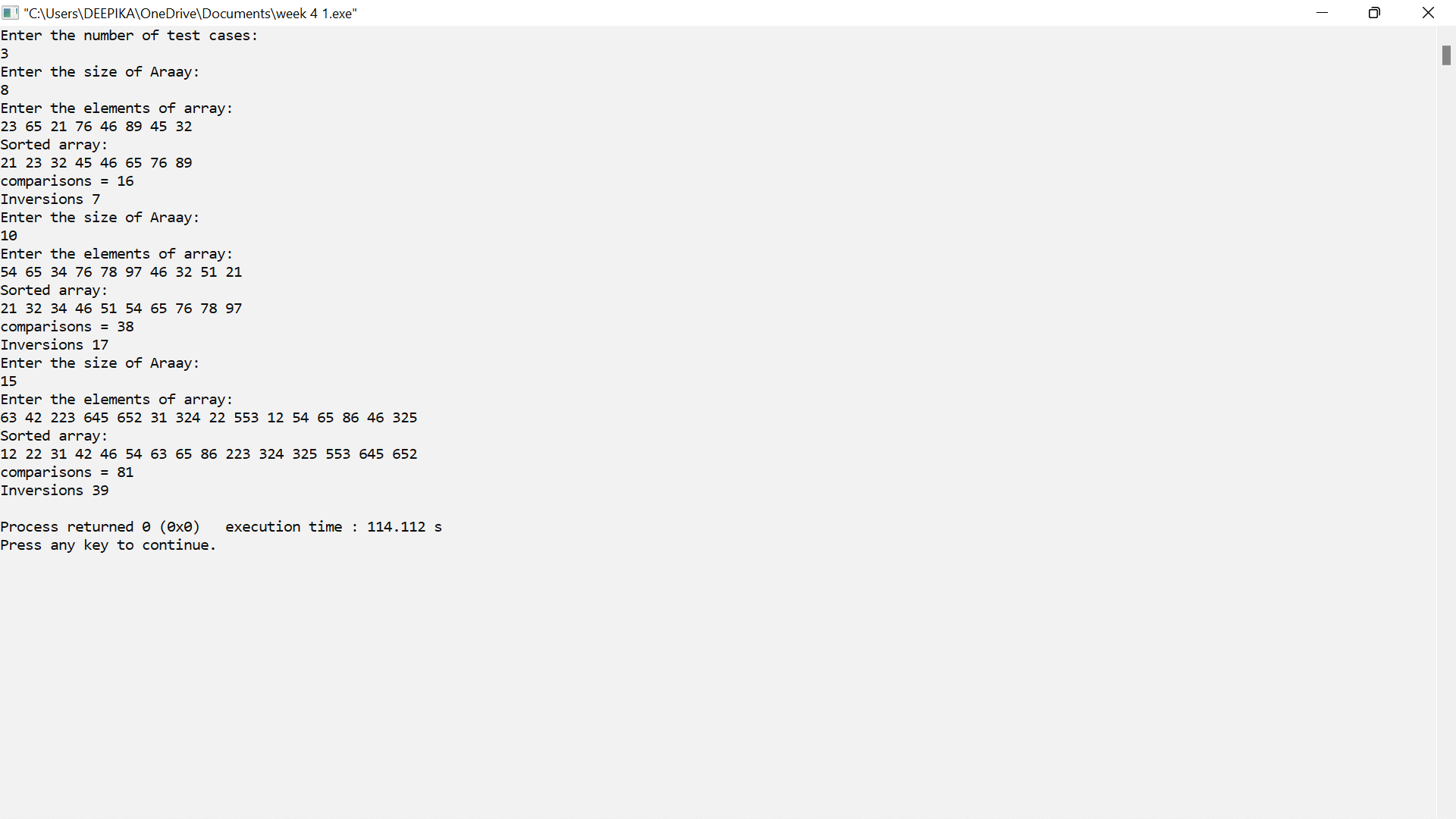
cout<<"comparisons = "<<comp<<endl;

cout<<"Inversions "<<inv<<endl;

}

}

**OUTPUT:**



**11.** **Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by partitioning the array into two subarrays based on a pivot element such that one of the sub array holds values smaller than the pivot element while another sub array holds values greater than the pivot element. Pivot element should be selected randomly from the array. Your program should also find number of comparisons and swaps required for sorting the array**

**11.QUICK SORTQUICK SORT**

**CODE:**

#include <bits/stdc++.h>

using namespace std;

int comp= 0, swaps = 0;

int partition(int arr[], int l, int h)

{

int x = (rand() % (l - h)) + l;

if(h != x)

{

//swaps++;

comp++;

swap(arr[x], arr[h]);

}

int pivot = arr[h];

int i = l - 1;

for(int j = l; j <= h - 1; j++)

{

if(arr[j] <= pivot) {

i++;

swaps++;

comp++;

swap(arr[i], arr[j]);

}

}

//swaps++;

swap(arr[i + 1], arr[h]);

return i + 1;

}

void quickSort(int arr[], int l, int h)

{

if(l < h)

{

int pivot = partition(arr, l, h);

quickSort(arr, l, pivot - 1);

quickSort(arr, pivot + 1, h);

}

}

void display(int arr[], int n)

{

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

{

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

}

cout<<endl;

}

int main()

{

int t;

cout<<"Enter the number of test cases:" <<endl;

cin>>t;

while(t--)

{

int n;

cout<<"Enter the size of Araay:"<<endl;

cin>>n;

int arr[n];

cout<<"Enter the elements of array:"<<endl;

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

{

cin>>arr[i];

}

quickSort(arr, 0, n - 1);

cout<<"Sorted array:"<<endl;

display(arr, n);

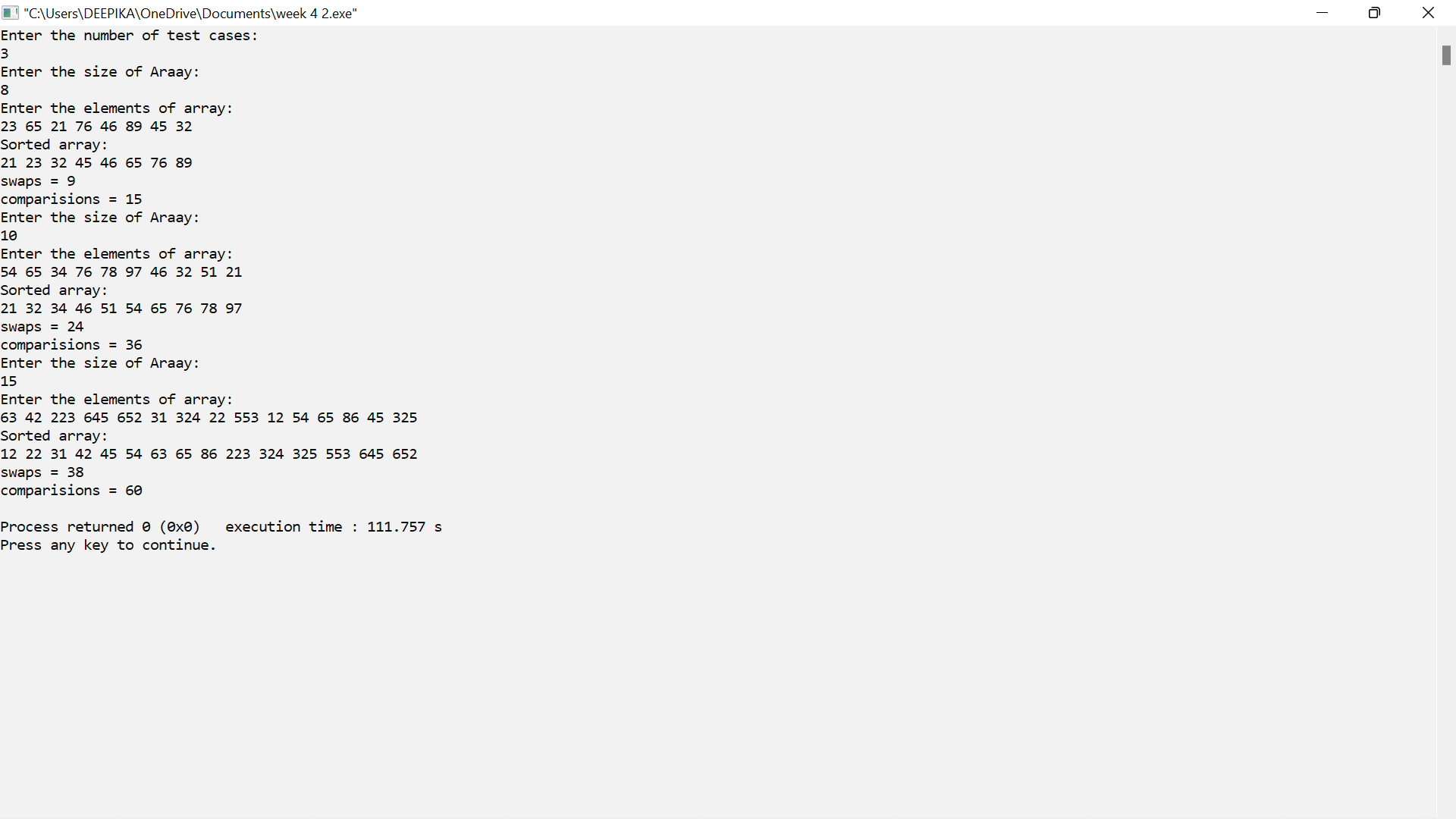
cout<<"swaps = "<<swaps<<endl;

cout<<"comparisions = "<<comp<<endl;

}

}

**OUTPUT:**



**12. Given an unsorted array of integers, design an algorithm and implement it using a program to find Kth smallest or largest element in the array. (Worst case Time Complexity = O(n))**

**TO FIND Kth LARGESST OR SMALLEST ELEMENT**

**CODE:**

#include <bits/stdc++.h>

using namespace std;

int kthSmallest(int a[], int n, int k)

{

sort(a, a + n);

return a[k - 1];

}

int kthlargest(int a[], int n, int k)

{

sort(a, a + n);

return a[n-k];

}

int main()

{

int n,t,k;

cout<<"Enter the number of test cases:" <<endl;

cin>>t;

while(t--)

{

cout<<"Enter the size of Araay:"<<endl;

cin>>n;

cout<<"Enter the elements of array:"<<endl;

int a[n];

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

{

cin>>a[i];

}

cout<<"enter the value of k"<<endl;

cin>>k;

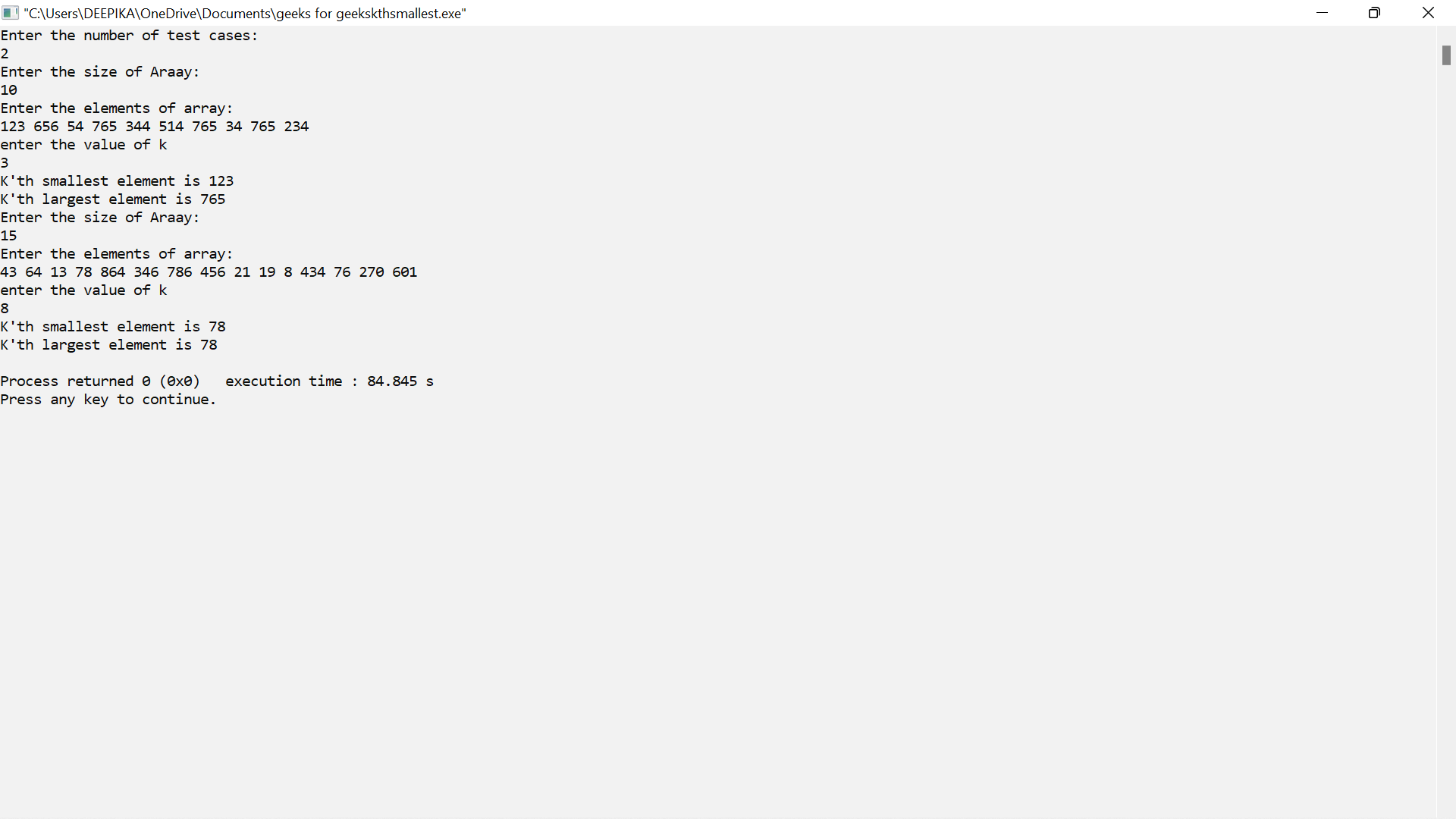
cout << "K'th smallest element is " << kthSmallest(a, n, k)<<endl;

cout << "K'th largest element is " << kthlargest(a, n, k)<<endl;

}

return 0;

}

**OUTPUT:** 

**13.** **Given an unsorted array of alphabets containing duplicate elements. Design an algorithm and implement it using a program to find which alphabet has maximum number of occurrences and print it. (Time Complexity = O(n)) (Hint: Use counting sort)**

**13.COUNT SORTCOUNT SORT**

**CODE:**

#include<iostream>

using namespace std;

int getmax(char a[],int n)

{

int max=a[0];

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

{

if(a[i]>max)

max=a[i];

}

return max;

}

void countsort(char a[],int n)

{

int output[n+1];

char max=getmax(a,n);

int count[max+1];

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

{

count[i]=0;

}

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

{

count[a[i]]++;

}

int f=0;

int g=count[0],i;

char ans;

for(i=1;i<max+1;i++)

{

if(count[i]>g)

{

f=1;

g=count[i];

ans=(char)i;

}

}

if(g==1)

cout<<"No Duplicates present"<<endl;

else

cout<<"max occurance character is "<<ans<<" and occurance is "<<g<<endl;

}

void printarr(char a[],int n)

{

int i;

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

cout<<a[i]<<endl;

}

int main()

{

int t,n;

cout<<"Enter the number of test cases:" <<endl;

cin>>t;

while(t--)

{

cout<<"Enter the size of Araay:"<<endl;

cin>>n;

cout<<"Enter the elements of array:"<<endl;

char a[n];

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

{

cin>>a[i];

}

int n=sizeof(a)/sizeof(a[0]);

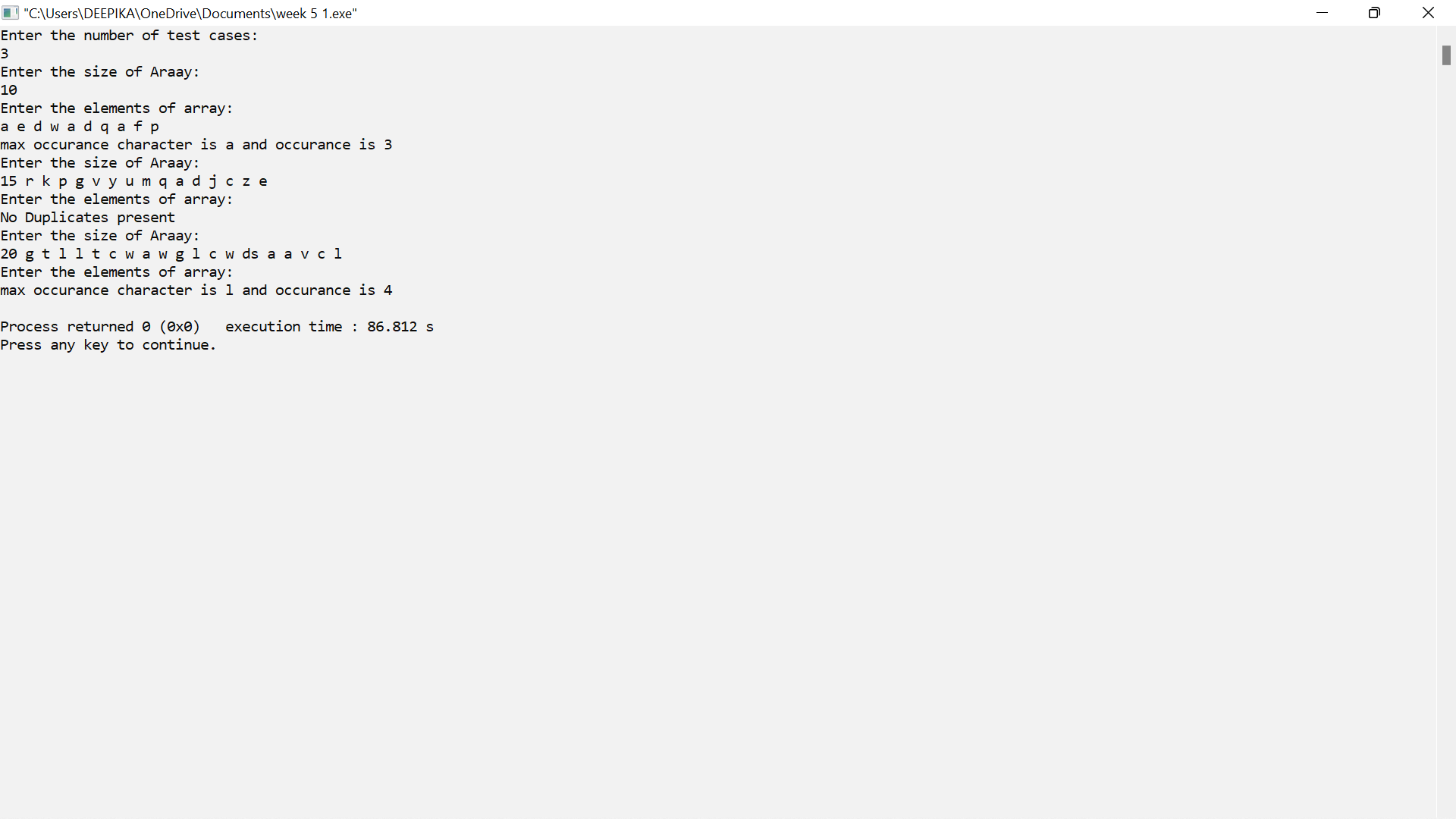
countsort(a,n);

}

return 0;

}

**OUTPUT:**



**14.** **Given an unsorted array of integers, design an algorithm and implement it using a program to find whether two elements exist such that their sum is equal to the given key element. (Time Complexity = O(n log n))**

**CODE:**

#include<bits/stdc++.h>

using namespace std;

int main()

{

int t;

cout<<"Enter the number of test cases:" <<endl;

cin>>t;

while(t--)

{

int n,i,j,key,flag=0;

cout<<"Enter the size of Araay:"<<endl;

cin>>n;

cout<<"Enter the elements of array:"<<endl;

int a[n];

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

{

cin>>a[i];

}

cout<<"Enter the key element"<<endl;

cin>>key;

sort(a,a+n);

i=0,j=n-1;

while(i<j)

{

int s=a[i]+a[j];

if(s==key)

{

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

flag=1;

i++;

j--;

}

else if(s<key)

i++;

else

j--;

}

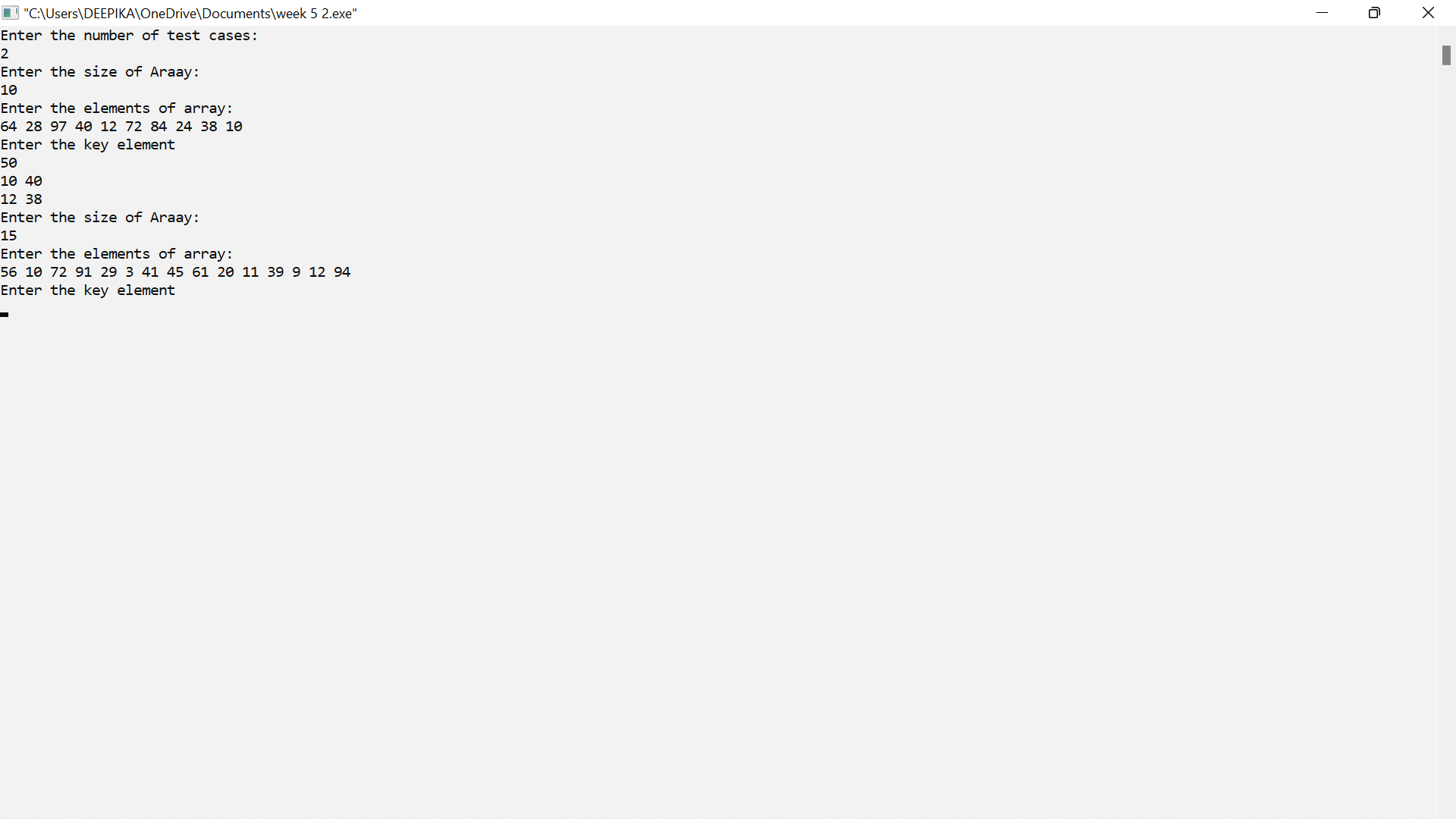
if(flag==0)

cout<<"No such elements"<<endl;

}

}

**OUTPUT:**



**15.** **You have been given two sorted integer arrays of size m and n. Design an algorithm and implement it using a program to find list of elements which are common to both. (Time Complexity = O(m+n))**

**CODE:**

#include <iostream>

using namespace std;

int main()

{

int n,m,i,j;

cout<<"Enter the size of 1st Araay:"<<endl;

cin>>n;

cout<<"Enter the elements of 1st array:"<<endl;

int a[n];

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

{

cin>>a[i];

}

cout<<"Enter the size of 2nd Araay:"<<endl;

cin>>m;

cout<<"Enter the elements of 2nd array:"<<endl;

int b[m];

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

{

cin>>a[j];

}

i=0,j=0;

while(i<n&&j<m)

{

if(a[i]==b[j])

{

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

i++;

j++;

}

else if(a[i]<b[j])

i++;

else

j++;

}

}

**OUTPUT:**

