**ASSIGNMENT-3**

**1.Implement the following searching techniques**

**a)Linear search**

**b)binary search**

**Linear search:**

**Pseudocode:**

Start

Declare int a to enter size of an array

Declare int arr[a],i,key,comp=0

Run loop from i=0 to i<a to enter and print elements of an array

Take user input key searching element

Run loop from i=0 to i<a

For every iteration comp=comp+1

If(a[i]==key)

Print key found at i+1 position

Print number of comparisions are comp

If found break the loop

If(i==size of an array)

Print key not found

Print number of comparisons are comp

end

**C program:**

#include<stdio.h>

void main()

{

int a;

printf(" enter size of array:");

scanf("%d",&a);

int arr[a],i,key,comp=0;

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

{

printf("enter %d element:",i);

scanf("%d",&arr[i]);

}

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

{

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

}

printf("\n");

printf("enter key:");

scanf("%d",&key);

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

{

comp=comp+1;

if (arr[i]==key)

{

printf("found at position %d\n",i+1);

printf("no.of comparisions %d\n",comp);

break;

}

}

if(i==a)

{

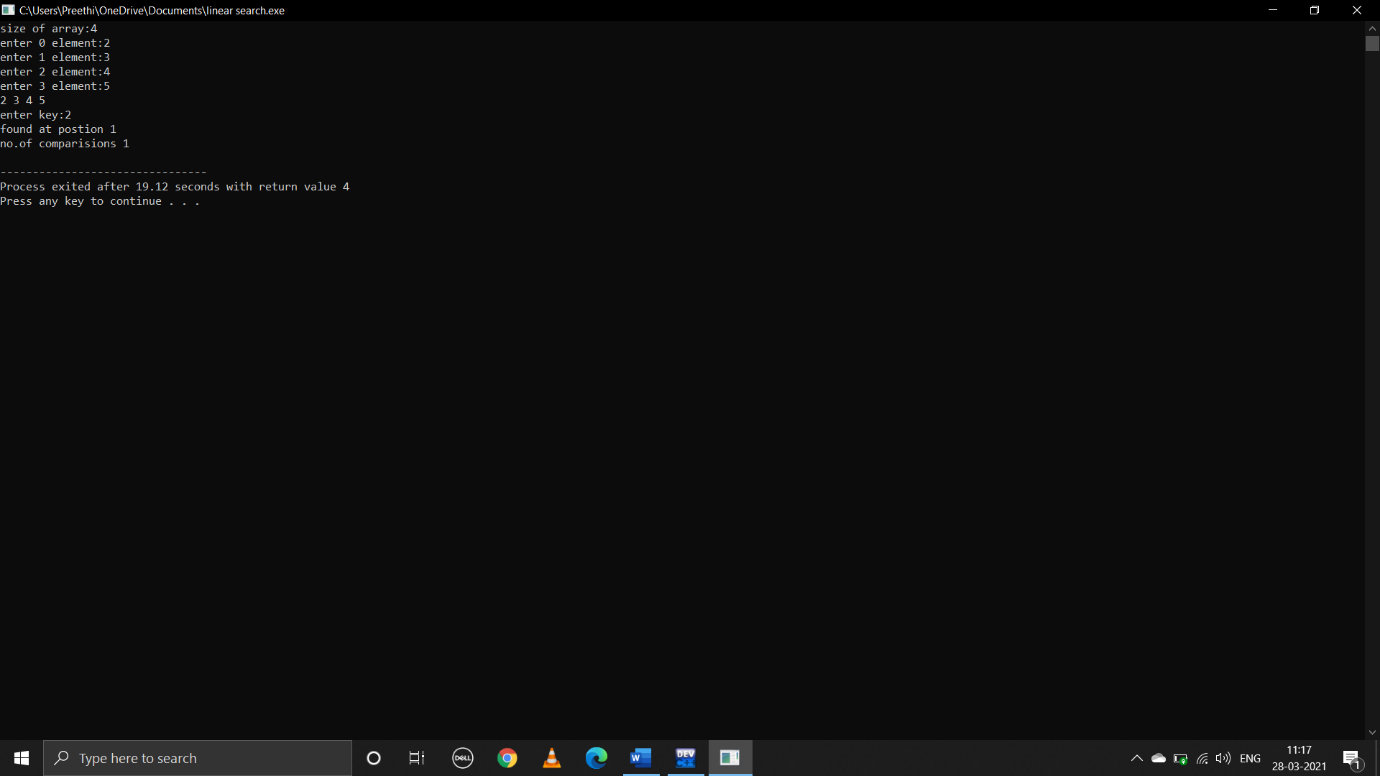
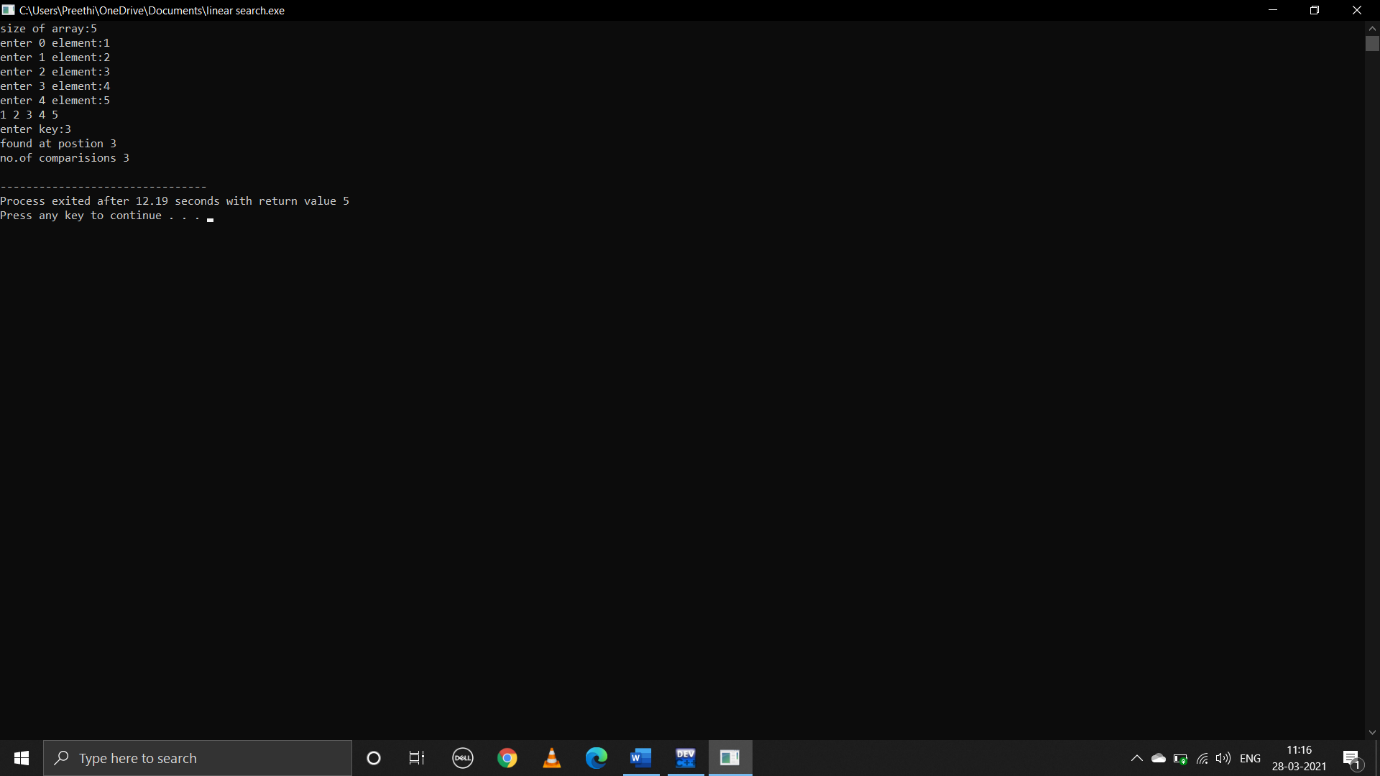
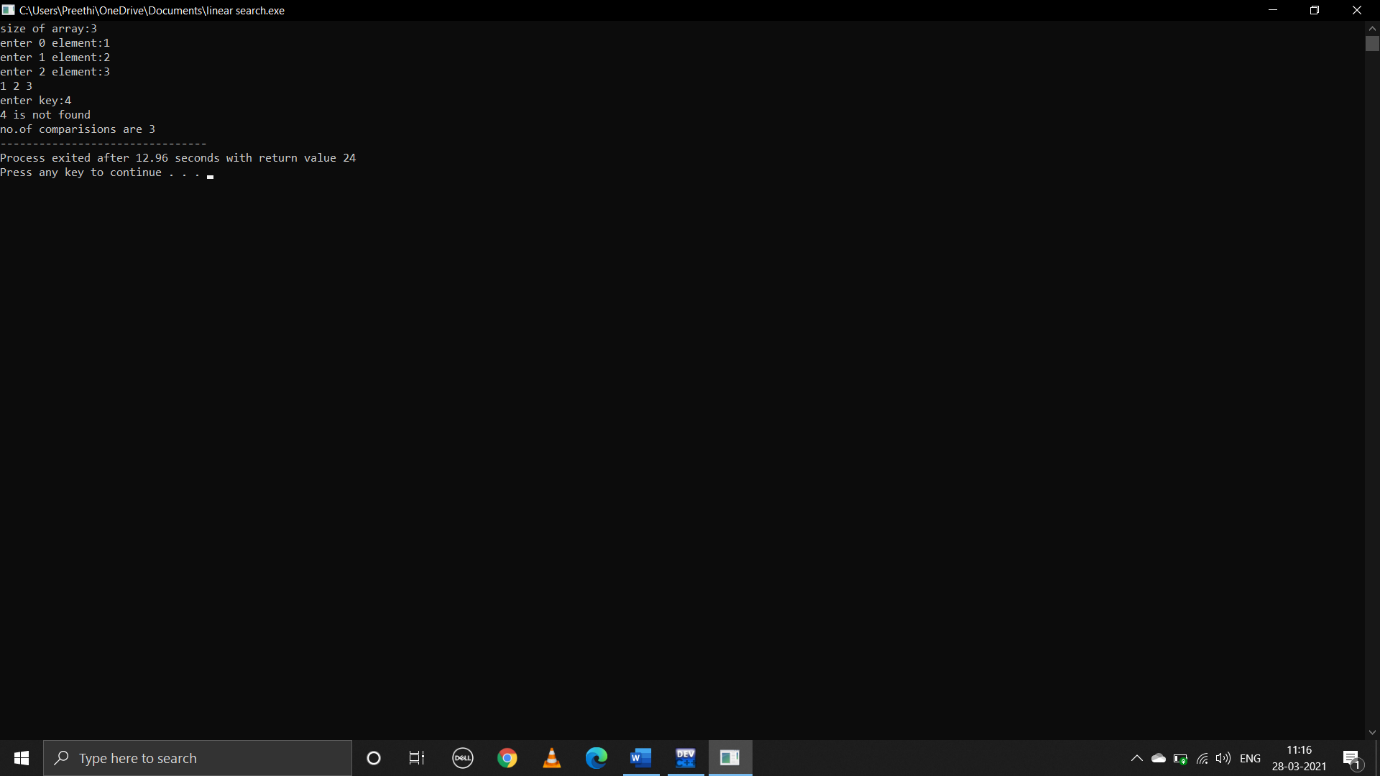
printf("%d is not found\n",key);

printf("no.of comparisions are %d",a);

}

}

**Output:**

****

**Binary search:**

**Pseudocode:**

Begin

Declare int a for size of an array

Declare int arr[a],key,first,last,middle

Run loop from i=0 to i<a to enter and print elements of an array

Take user input key searching element

Declare first=0,last=a-1,middle=(first+last)/2

While(first<=last)

If(arr[middle]<key)

First=middle+1,since array is sorted we need to search for key at right side of middle element

arr[middle]==key

print key found at middle index

break

else last=middle-1,we need to search for key at left side of middle element

if(first>last)

print(key not found)

end

**C program:**

#include<stdio.h>

void main()

{

int a;

printf("enter size of array:");

scanf("%d",&a);

int arr[a],i,key,first,last,middle;

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

{

printf("enter %d element:",i);

scanf("%d",&arr[i]);

}

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

{

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

}

printf("enter key:");

scanf("%d",&key);

first=0;

last=a-1;

middle=(first+last)/2;

while(first<=last)

{

if(arr[middle]<key)

first=middle+1;

else if(arr[middle]==key)

{

printf("key found at index %d",middle);

break;

}

else

last=middle-1;

middle=(first+last)/2;

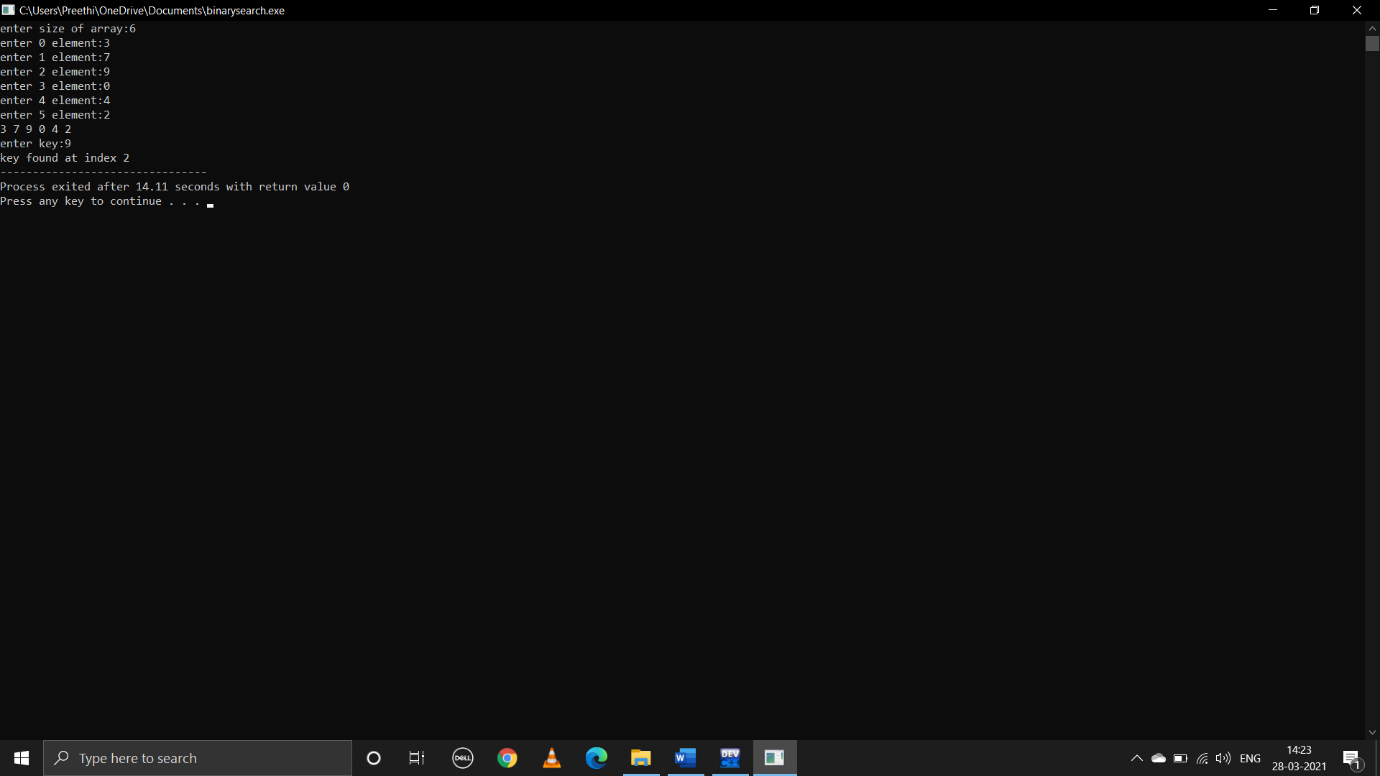
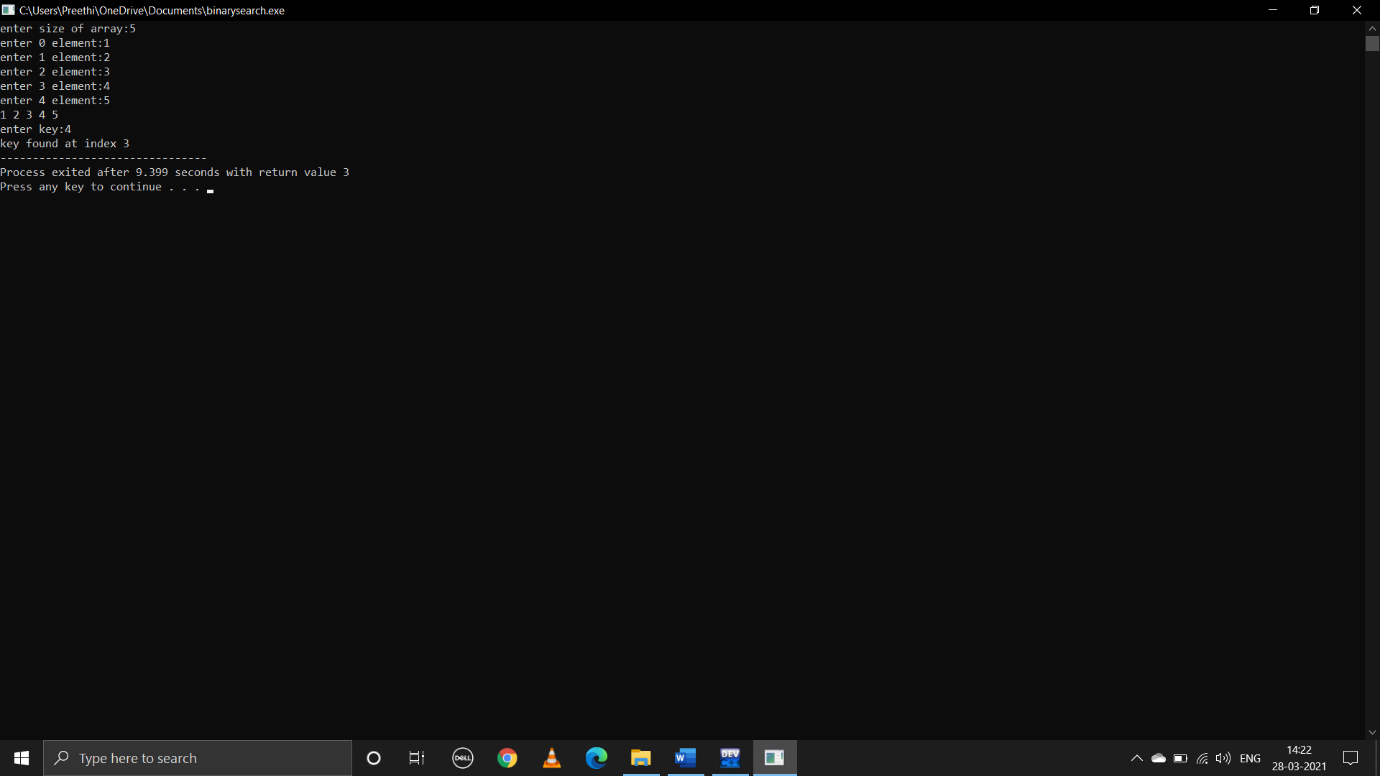
}

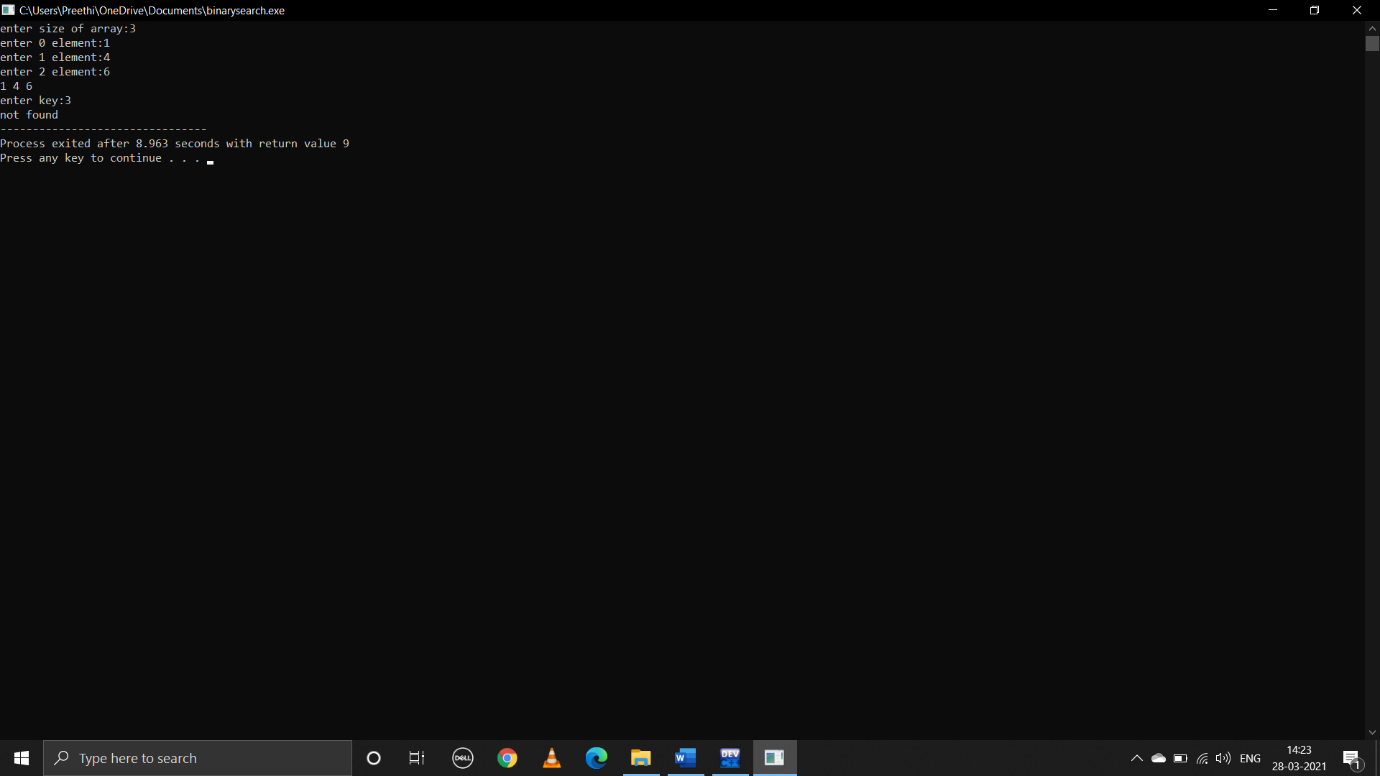
if(first>last)

printf("not found");

}

**Output:**

****

****

**2.Implement the following sorting techniques**

**a)bubble sort**

**b)selection sort**

**c)insertion sort**

**d)quick sort**

**e)merge sort**

**a)bubble sort:**

**pseudocode:**

begin

take user input size of an array n

int a[n]

run loop from i=0 to i<size to print and enter elements of an array

now,to sort this user input array call the function bubblesort(a,n)

declare functions: bubblesort(int arr[], int size) and printarray(int arr[], int size)

bubblesort function:

int i,j,temp

run loops two times:one to run through the array and other is for comparision

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

for(j=0;j<size-1;j++)

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

swap j+1 and j

to print array status after each pass call the function printarray which prints array

end

**c program:**

#include<stdio.h>

void bubblesort(int arr[],int size);

main()

{

int n,i;

printf("enter size of an array:");

scanf("%d",&n);

int a[n];

printf("enter elements in array:\n");

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

{

scanf("%d",&a[i]);

}

printf("unsorted array is:\n");

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

{

printf(" %d ",a[i]);

}

printf("\n");

bubblesort(a,n);

}

void bubblesort(int arr[],int size)

{

int temp,i,j;

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

{

for(j=0;j<size-1;j++)

{

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

{

temp=arr[j];

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

arr[j+1]=temp;

}

}

printf("array status after pass %d\n",i);

printarray(arr,size);

printf("\n");

}

}

void printarray(int arr[],int size)

{

int i;

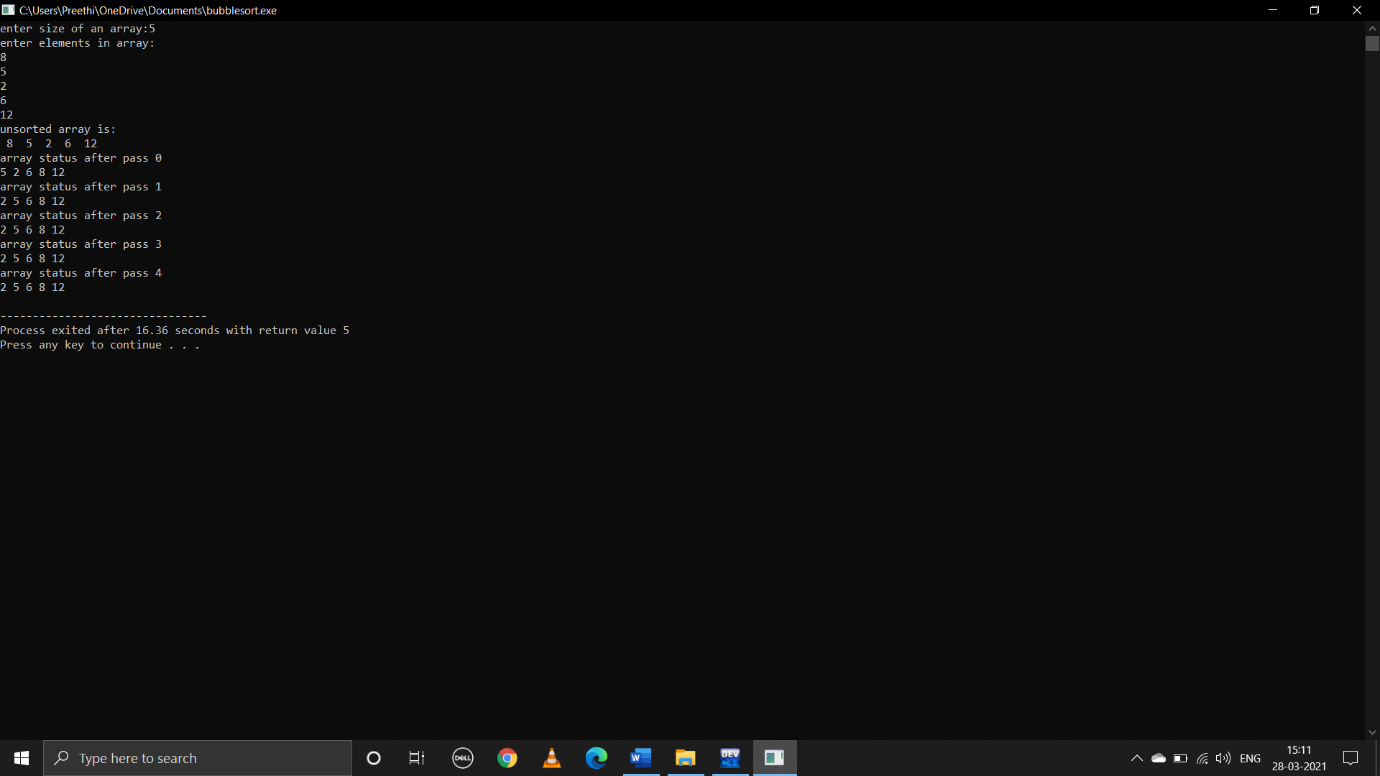
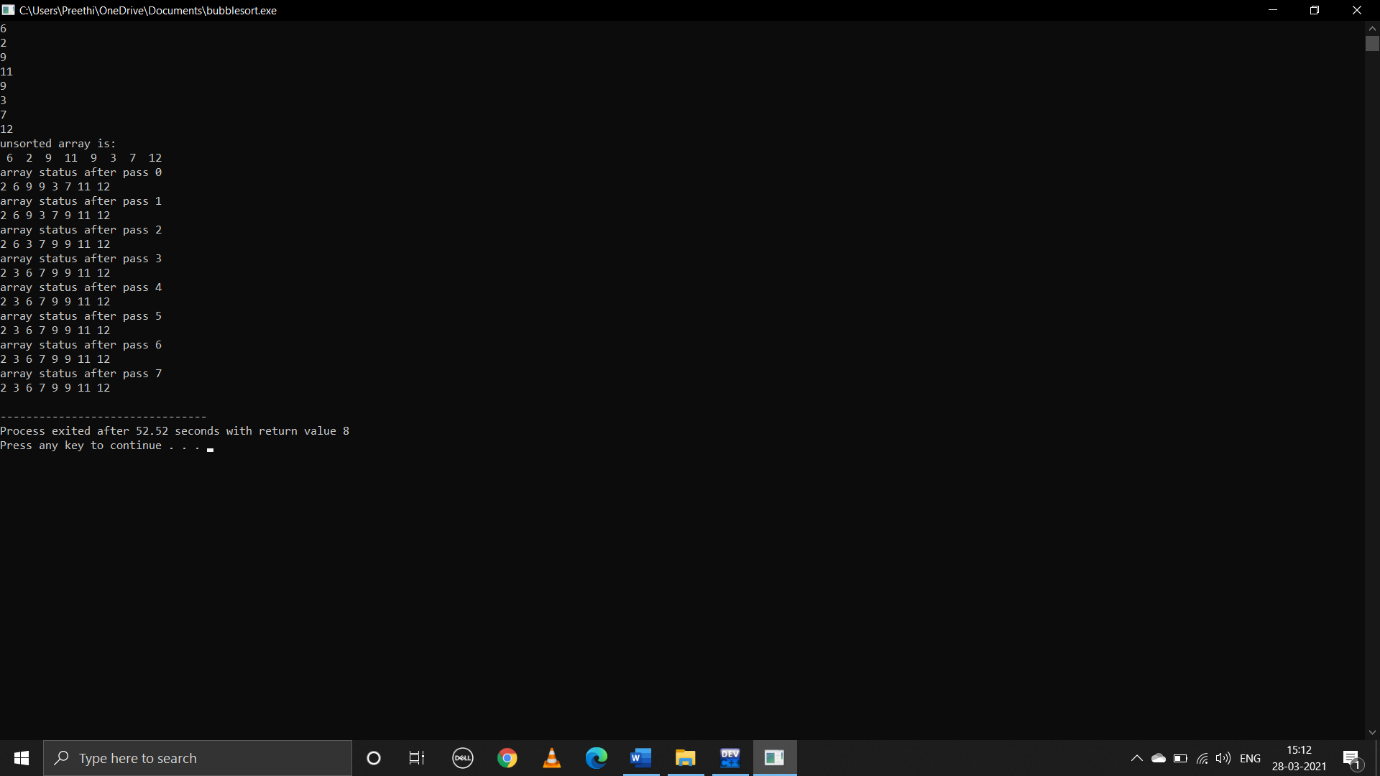
for(i=0;i<size;i++)

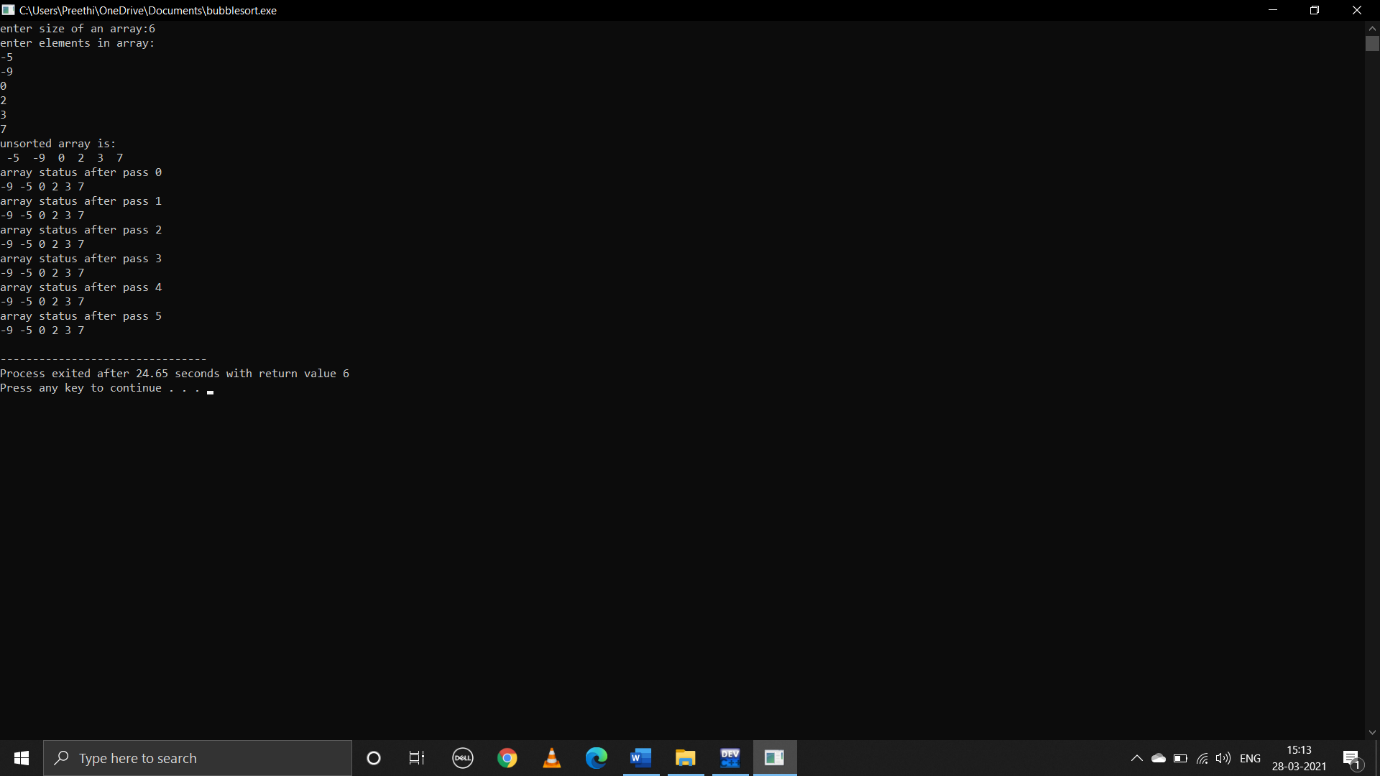
{

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

}

}

**Output: **

****

**b)selection sort:**

**pseudocode:**

begin

take user input size of an array

run loop from int i=0 to i<size to print and enter the elements in to the array

to sort this unsorted userinput list call the function selectionsort(int a[],int size)

now control jumps to the selection sort function definition

declare another two functions one is to print array another is to give maximum element index

now run loop from size-1 to 1

for(int i=size-1;i>=1;i--)

to place the greatest element at last

swap a[i] with a[max]

to print array status after each pass call the function print array

end

**c program:**

#include<stdio.h>

int selectionsort(int a[],int size);

int max(int a[],int size);

main()

{

int i,n;

printf("enter size of an array:");

scanf("%d",&n);

int arr[n];

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

{

scanf("%d",&arr[i]);

}

printf("unsorted array is:\n");

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

{

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

}

printf("\n");

selectionsort(arr,n);

}

void printarray(int a[],int size)

{

int i;

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

printf("%d ",a[i]);

}

int max(int a[],int size)

{

int max=0,i;

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

{

if(a[i]>a[max])

{

max=i;

}

}

return(max);

}

int selectionsort(int a[],int size)

{

int temp,big,i;

for(i=size-1;i>=1;i--)

{

big=max(a,i);

temp=a[big];

a[big]=a[i];

a[i]=temp;

printf("\n");

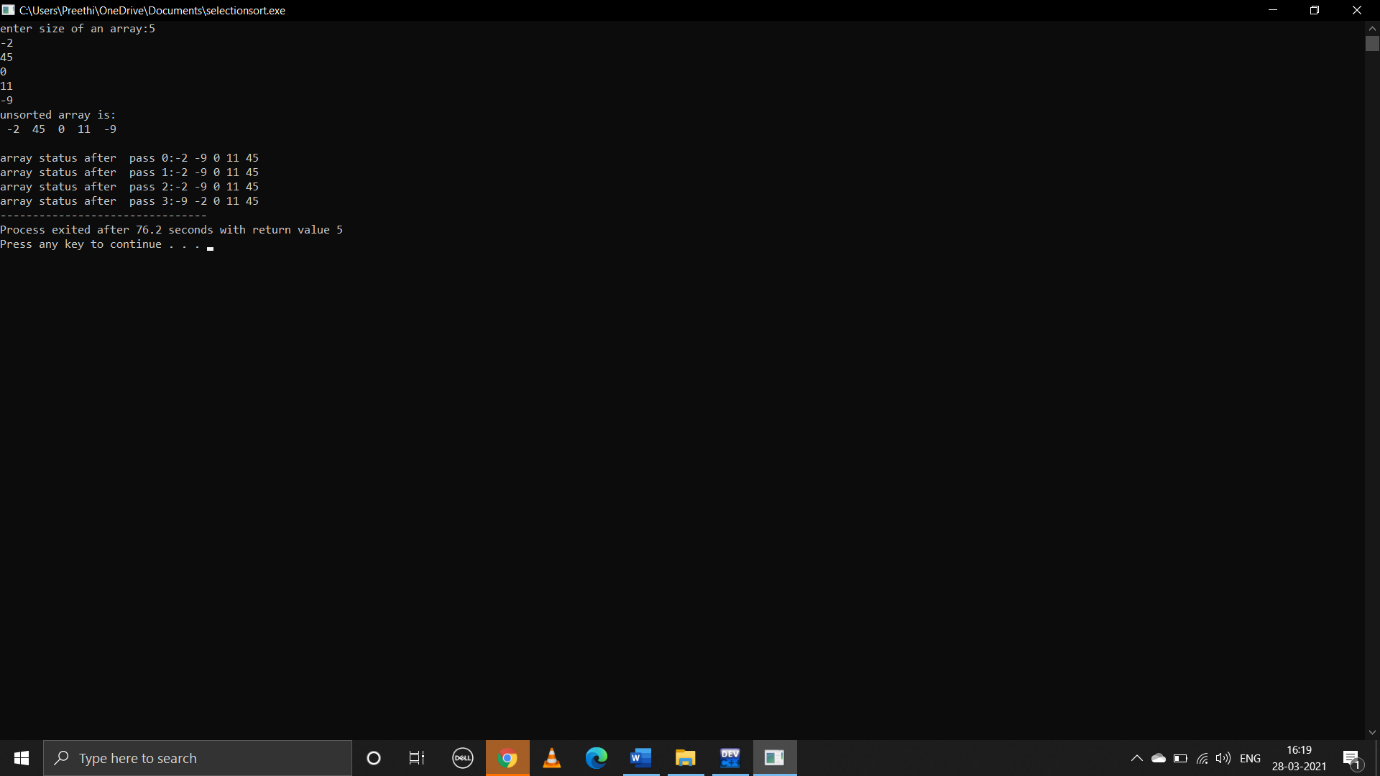
printf("array status after pass %d:",size-i-1);

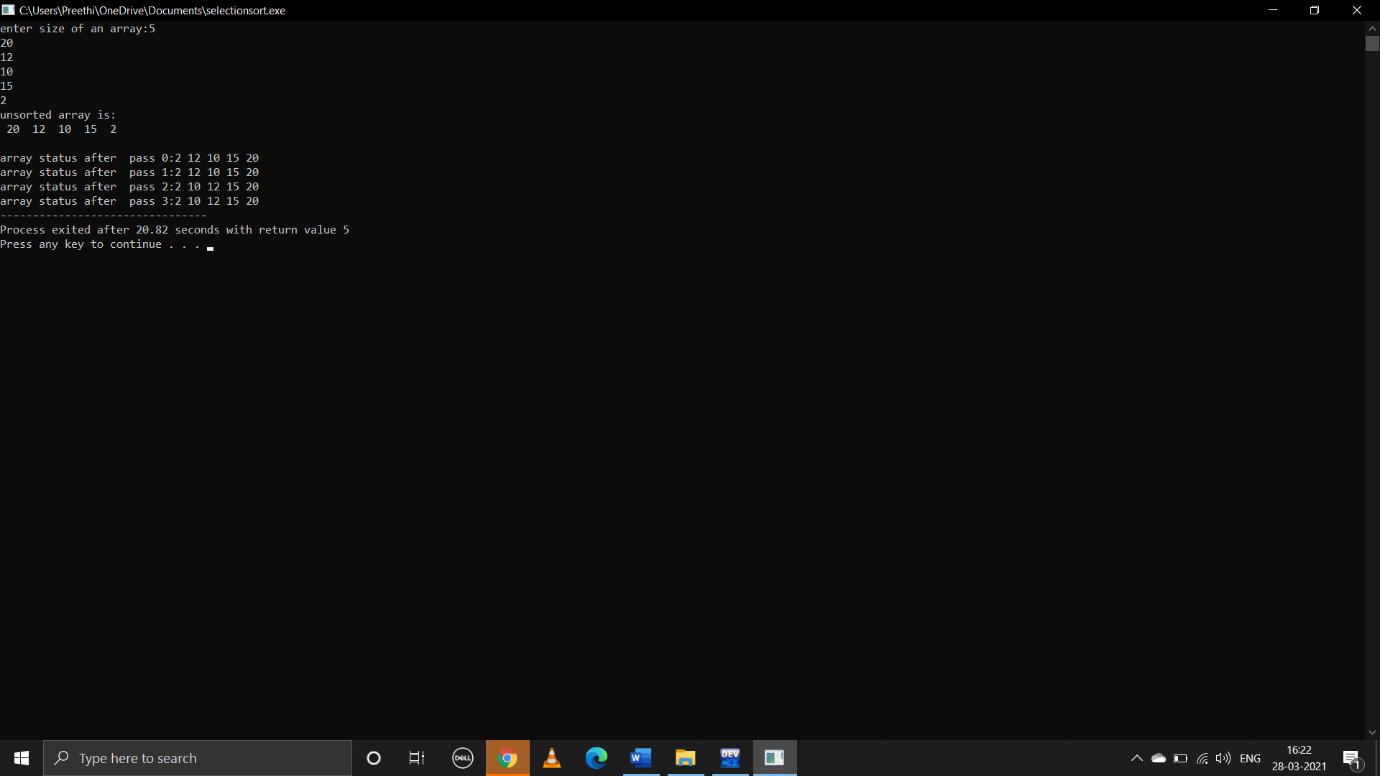
printarray(a,size);

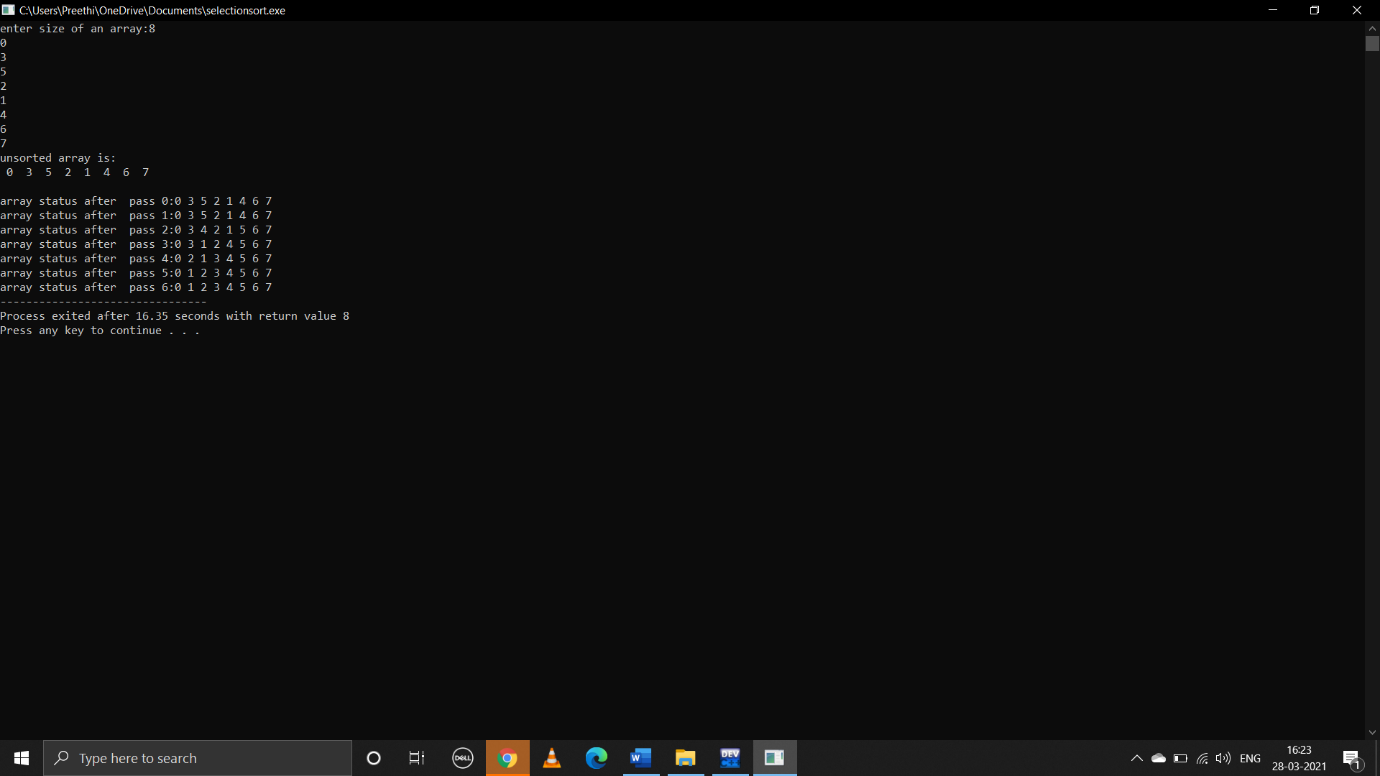
}

return;

}

**Output: **

****

****

**c)insertion sort:**

**pseudocode:**

begin

take user input for size of an array

using loop from i=0 to size of an array to enter and print elements of an array

to sort this unsorted user input array call the function insertion sort

control jumps to the function definition insertionsort(int a[],int size)

declare I,j,temp;

consider 1st element in array is sorted consider it as temp

now,compare temp with other elements in array if temp is greater than any other element in array

place that element behind the temp

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

temp=a[i]

for(j=I;j>0&&a[j-1]>temp;j--)

a[j]=a[j-1]

a[j]=temp

to print array status after each pass call the function printarray which prints the array

end

**c program:**

#include<stdio.h>

void insertionsort(int a[],int size);

main()

{

int n,i;

printf("enter size of an array:");

scanf("%d",&n);

int arr[n];

printf("enter elements in the array:\n");

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

{

scanf("%d",&arr[i]);

}

printf("unsorted array is:\n");

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

{

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

}

insertionsort(arr,n);

}

void insertionsort(int a[],int size)

{

int i,j,temp;

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

{

temp=a[i];

for(j=i;j>0&&a[j-1]>temp;j--)

{

a[j]=a[j-1];

}

a[j]=temp;

printf("\n");

printf("array status after pass %d:\n",i-1);

printarray(a,size);

}

}

void printarray(int a[],int size)

{

int i;

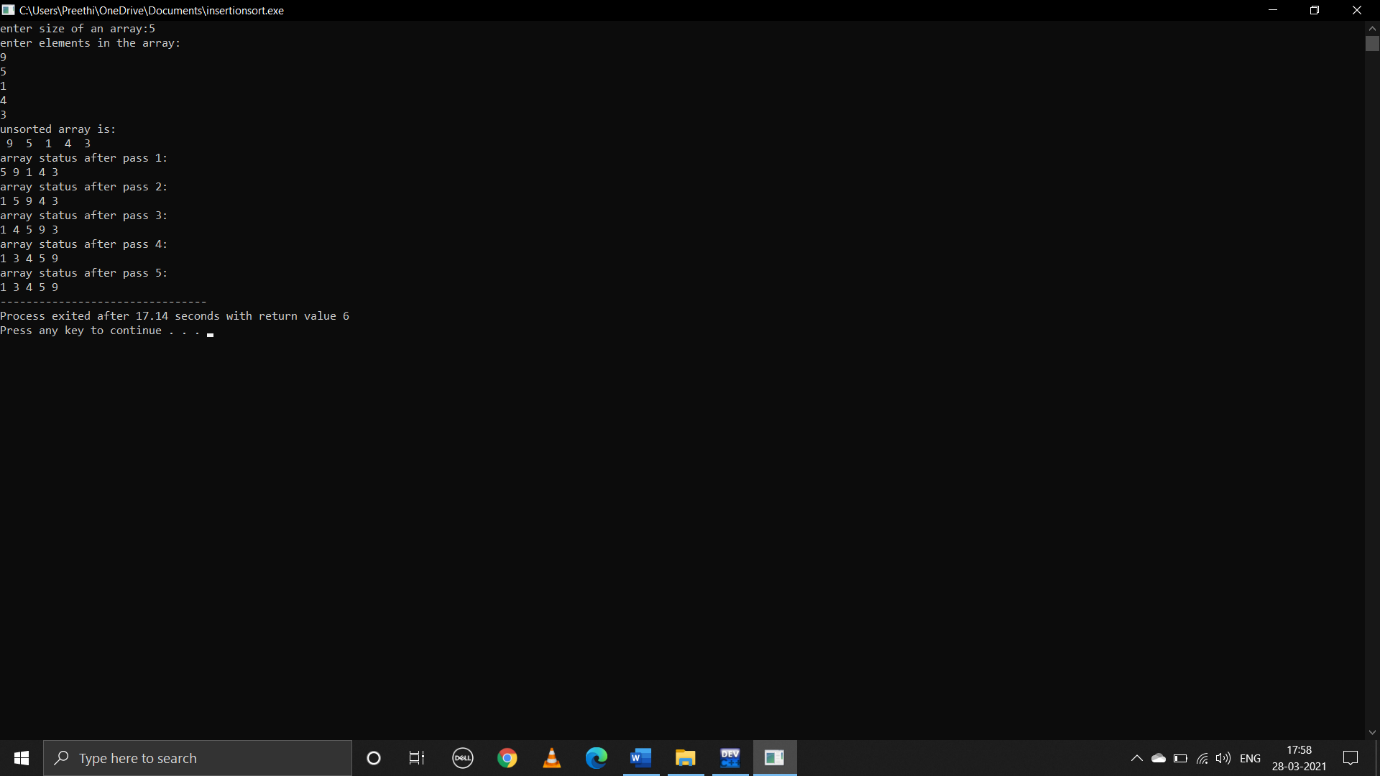
for(i=0;i<size;i++)

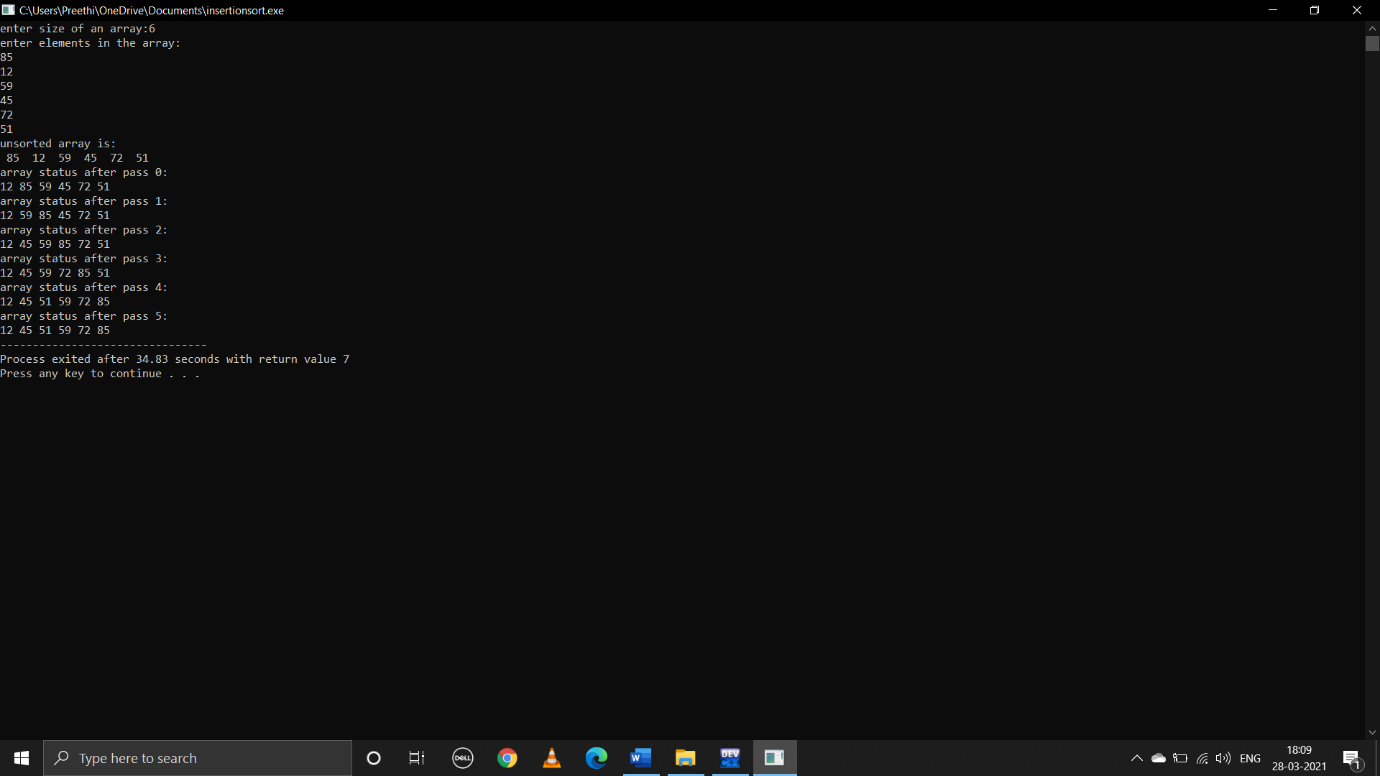
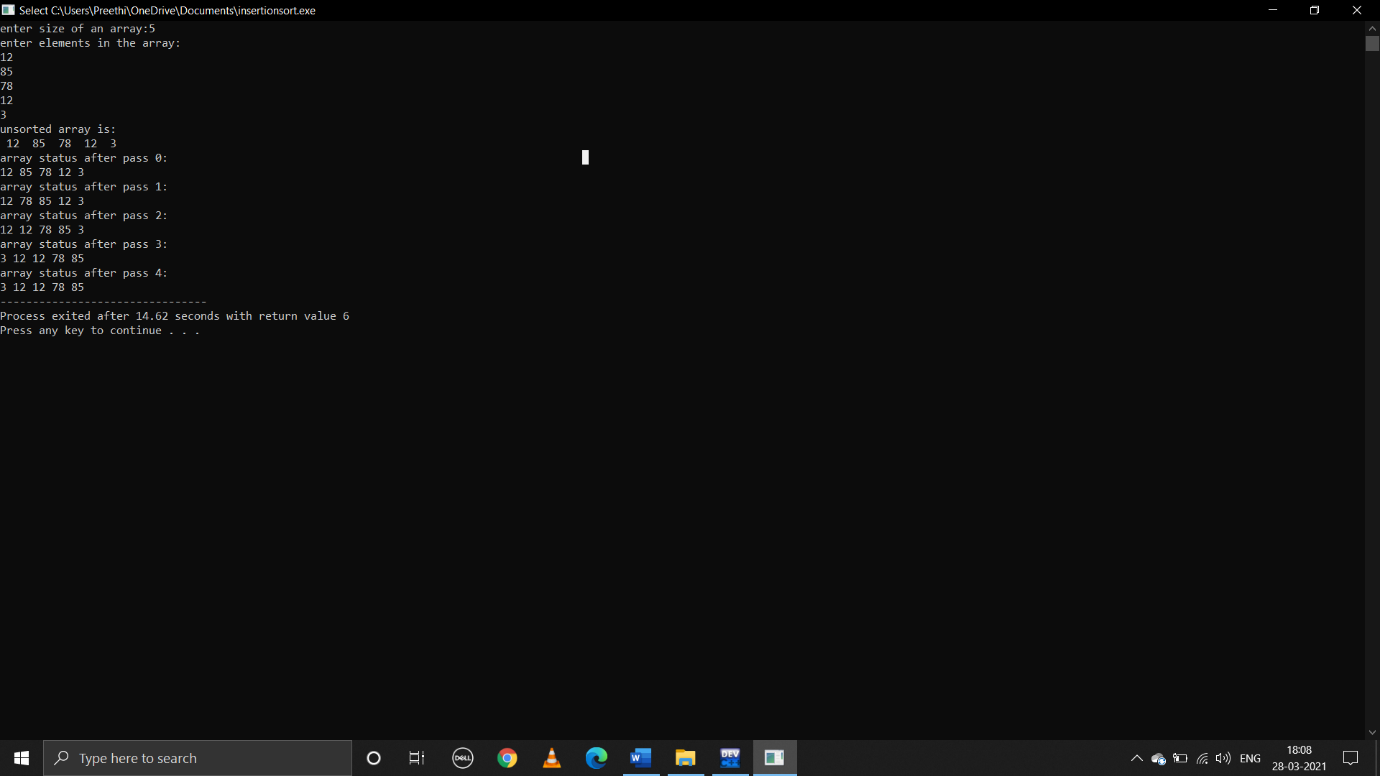
{

printf("%d ",a[i]);

}

}

**Output: **

****

**d)quicksort:**

**pseudocode:**

begin

declare int i,n,a[]

take user input array of size n

call the function quicksort

control jumps to the function definition quicksort

in this function declare int i,j,pivot ,temp

first element =pivot

i=first

j=last

while(i<j)

while(a[i]<=a[pivot]&&i<last)

{ i++;

while(a[j]>a[pivot])

j--;

if(i<j)

swap a[i] and a[j]

}

Swap a[j] and a[pivot]

Now pivot element is in its correct position

Call printarray function which prints array to print array status after each pass

Printarray(arr,size)

{

Declare static int i=0 to store value of last execution

Declare static int i=0 to store value of last execution

Print i

i++

run loop from int j=0 to j<size

{

Print arr[i]

}

Call quicksort function to sort elements on right and left side of pivot

quicksort(a,0,j-1,n)

quicksort(a,j+1,last,n)

end

**c program:**

#include<stdio.h>

void quicksort(int a[],int first,int last,int n)

{

int i,j,pivot,temp;

if(first<last)

{

pivot=first;

i=first;

j=last;

while(i<j)

{

while(a[i]<=a[pivot]&&i<last)

i++;

while(a[j]>a[pivot])

j--;

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

temp=a[pivot];

a[pivot]=a[j];

a[j]=temp;

printarray(a,n);

printf("\n");

quicksort(a,0,j-1,n);

quicksort(a,j+1,last,n);

}

}

int main()

{

int n,i;

printf("enter size of array:");

scanf("%d",&n);

int a[n];

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

{

scanf("%d",&a[i]);

}

printf("before sorted:\n");

printarray(a,n);

quicksort(a,0,n-1,n);

printf("after sorted:\n");

printarray(a,n);

return 0;

}

void printarray(int a[],int last)

{

static int i=0;

printf("array status after pass %d:",i);

i++;

int j;

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

{

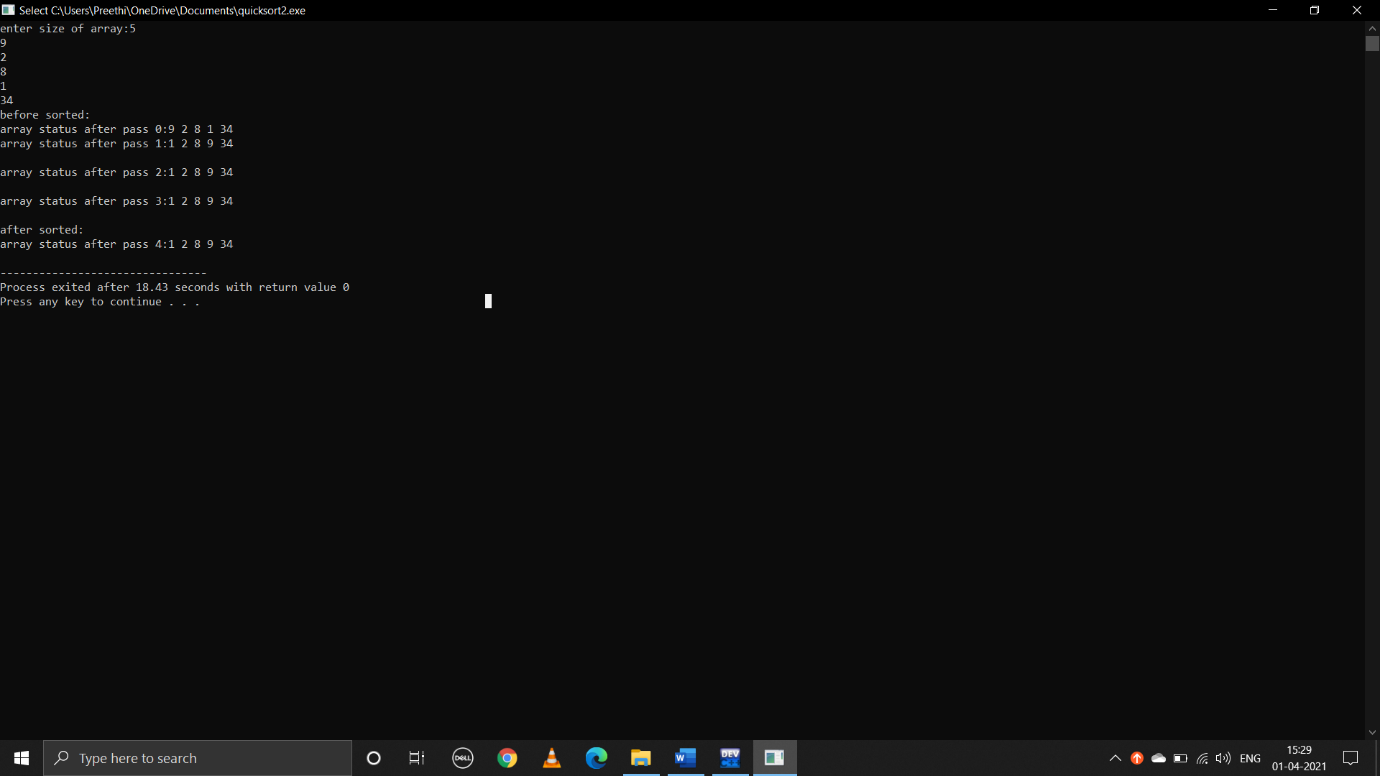
printf("%d ",a[j]);

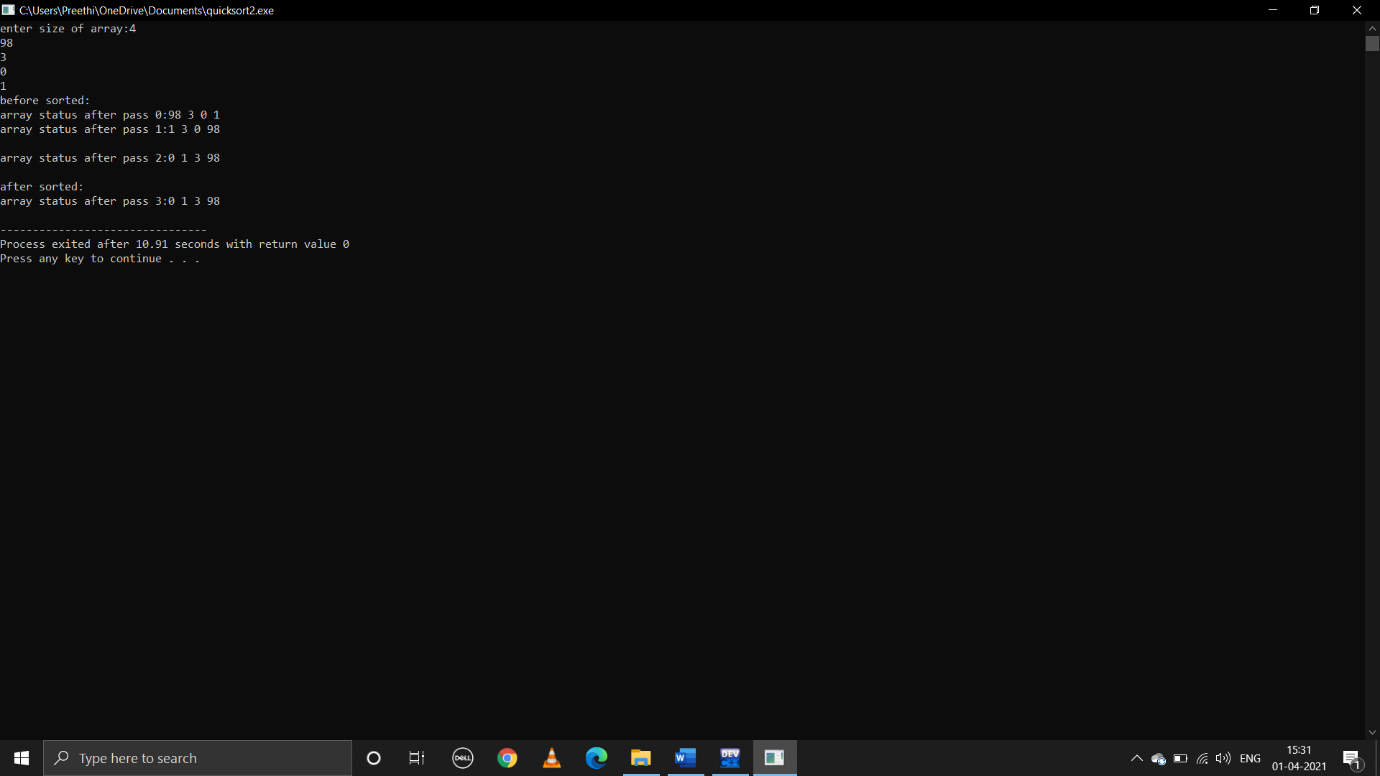
}

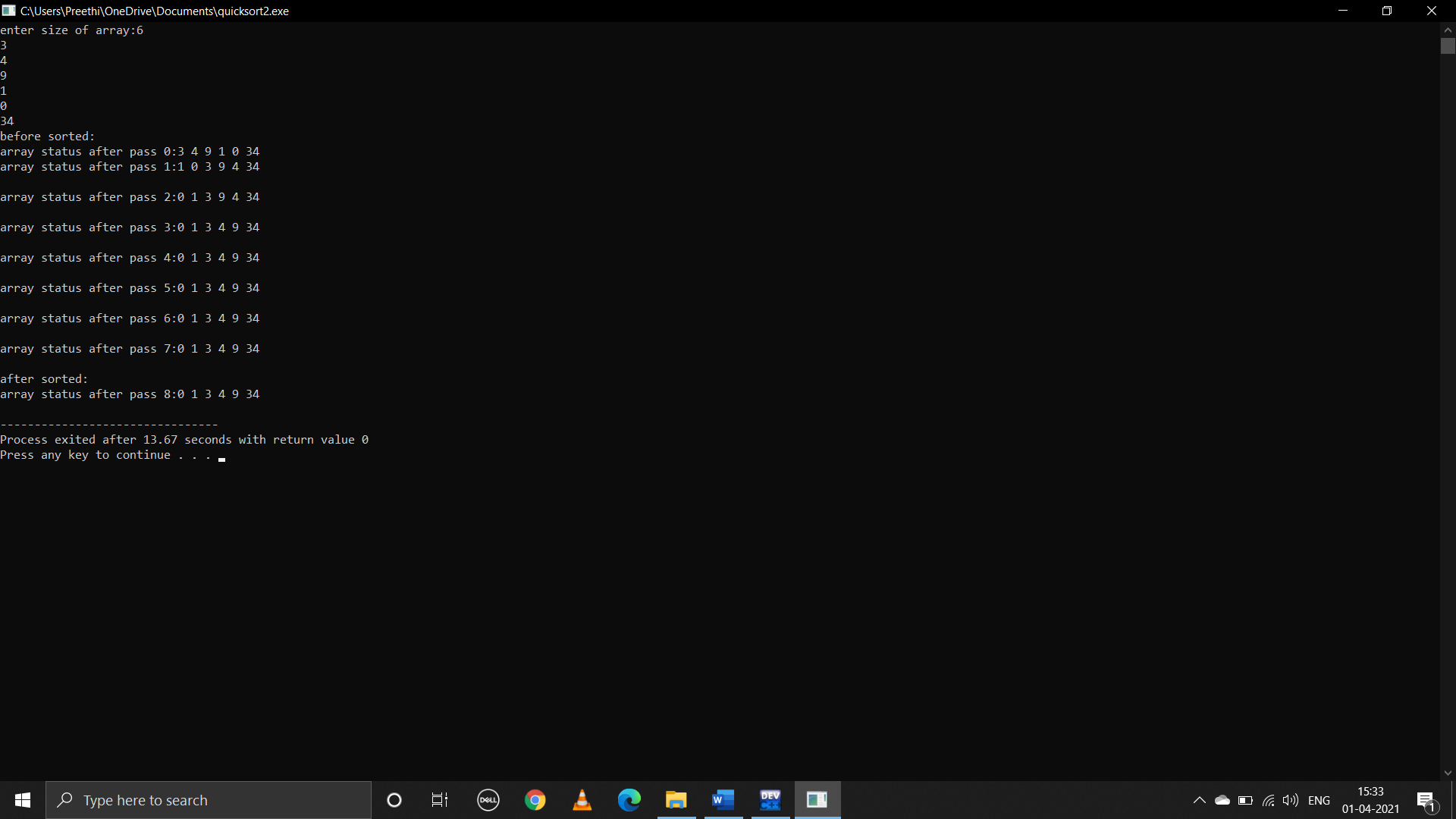
printf("\n");

}

**Output:**

****

****

****

**Merge sort:**

**Pseudocode:**

Begin

Take user input array of size n

Declare left and right variables which will mark the extreme indices of the array

Left will be assigned to 0 right will be assigned to n-1

For sorting this call the function mergesort function

Now control jumps to the mergesort function

Find mid=(left+right)/2

Call mergesort on(left,mid,n) and(mid+1,last,n)

Above will continue till left<right

Then we will call merge on the 2 subproblems

merge(arr,p,q,r,n)

n1=[p-q+1]

n2=[r-q]

int l[n1]

int m[n2]

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

l[i]=arr[p+i]

for(j=0;j<n2;j++)

m[j]=arr[q+1+j]

k=p

l and m have elements

while(i<n1 && j<n2)

{

if(l[i]<=m[j])

we are merging elements in ascending to arr[k]

add l[i] to arr[k]

i++

else add m[j] to arr[k]

j++

k++

}

While l has elements

While(i<n1)

{

arr[k]=l[i]

i++

k++

}

While m has elements

While(j<n2)

{

arr[k]=m[j]

j++

k++

}

Call the function print array which prints array after each pass

Printarray(arr,size)

{

Declare static int i=0 to store value of last execution

Print i

i++

run loop from int j=0 to j<size

{

Print arr[i]

}

End;

**C program:**

#include<stdio.h>

void merge(int arr[],int p,int q,int r,int n)

{

int n1=q-p+1;

int n2=r-q;

int l[n1],m[n2];

int i,j,k;

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

{

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

}

for(j=0;j<n2;j++)

{

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

}

i=0;

j=0;

k=p;

while(i<n1 && j<n2)

{

if(l[i]<=m[j])

{

arr[k]=l[i];

i++;

}

else

{

arr[k]=m[j];

j++;

}

k++;

}

while(i<n1)

{

arr[k]=l[i];

i++;

k++;

}

while(j<n2)

{

arr[k]=m[j];

j++;

k++;

}

printarray(arr,n);

}

void mergesort(int arr[],int l,int r,int n)

{

if(l<r)

{

int m=(l+r)/2;

mergesort(arr,l,m,n);

mergesort(arr,m+1,r,n);

merge(arr,l,m,r,n);

}

}

void printarray(int arr[],int size)

{

static int i=0;

printf("array status after pass %d:",i);

i++;

int j;

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

{

printf("%d ",arr[j]);

}

printf("\n");

}

int main()

{

int n;

printf("enter size of an array:");

scanf("%d",&n);

int i,arr[n];

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

{

scanf("%d",&arr[i]);

}

printf("unsorted array is:\n");

printarray(arr,n);

mergesort(arr,0,n-1,n);

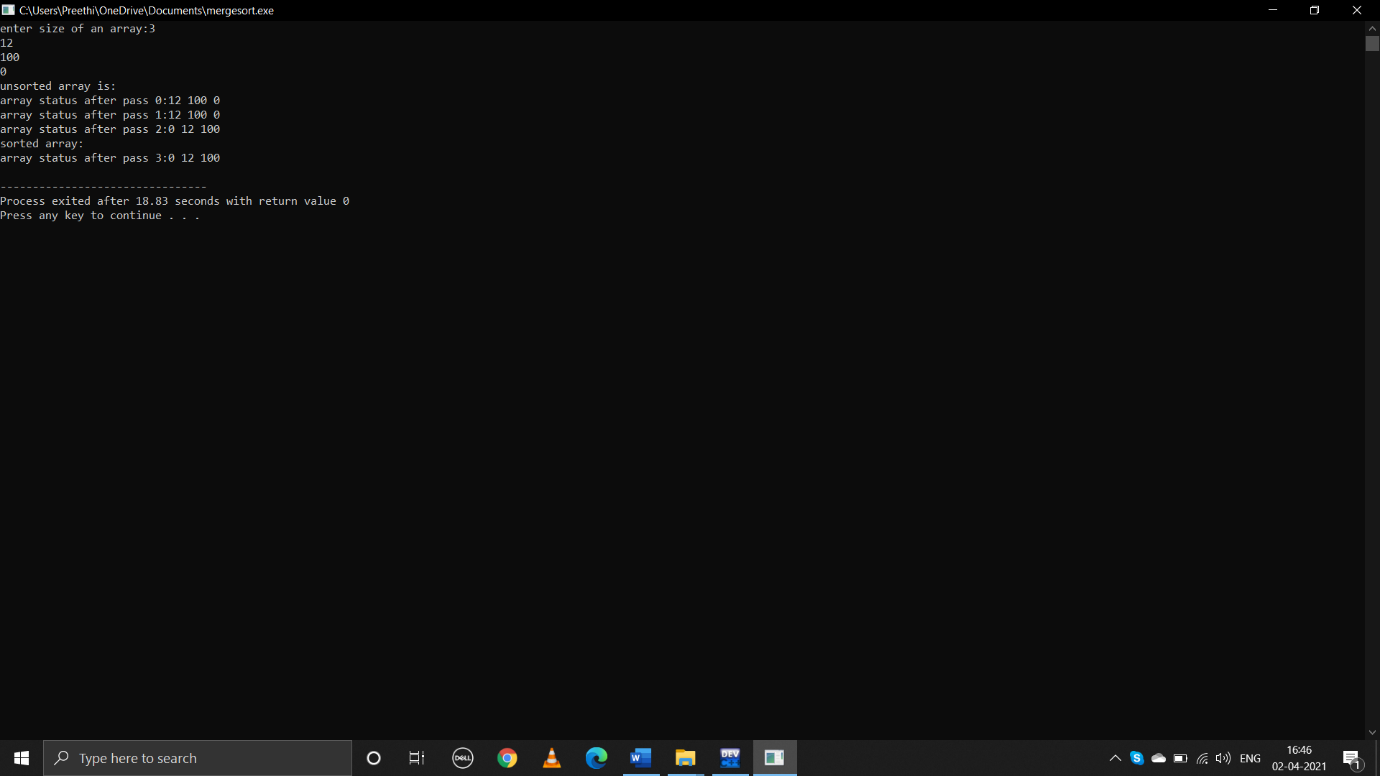
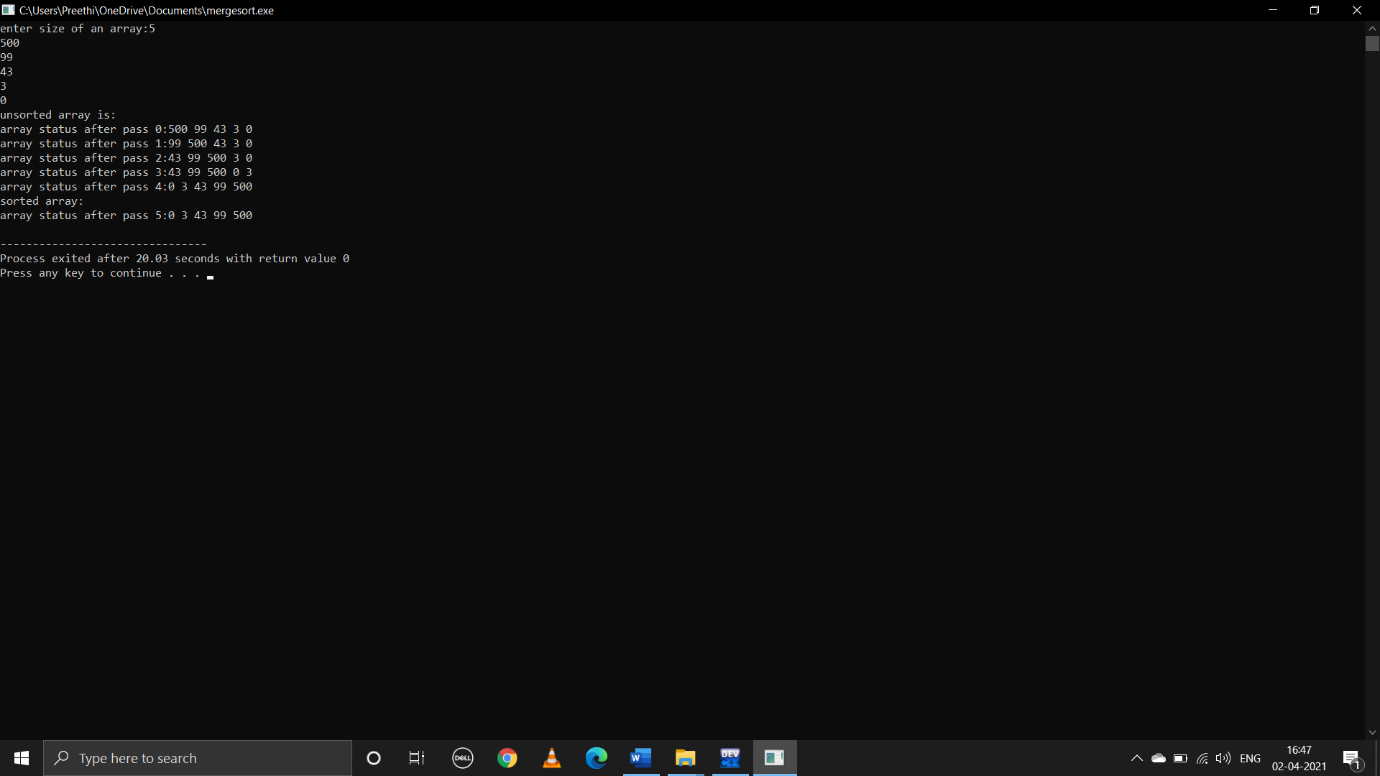
printf("sorted array:\n");

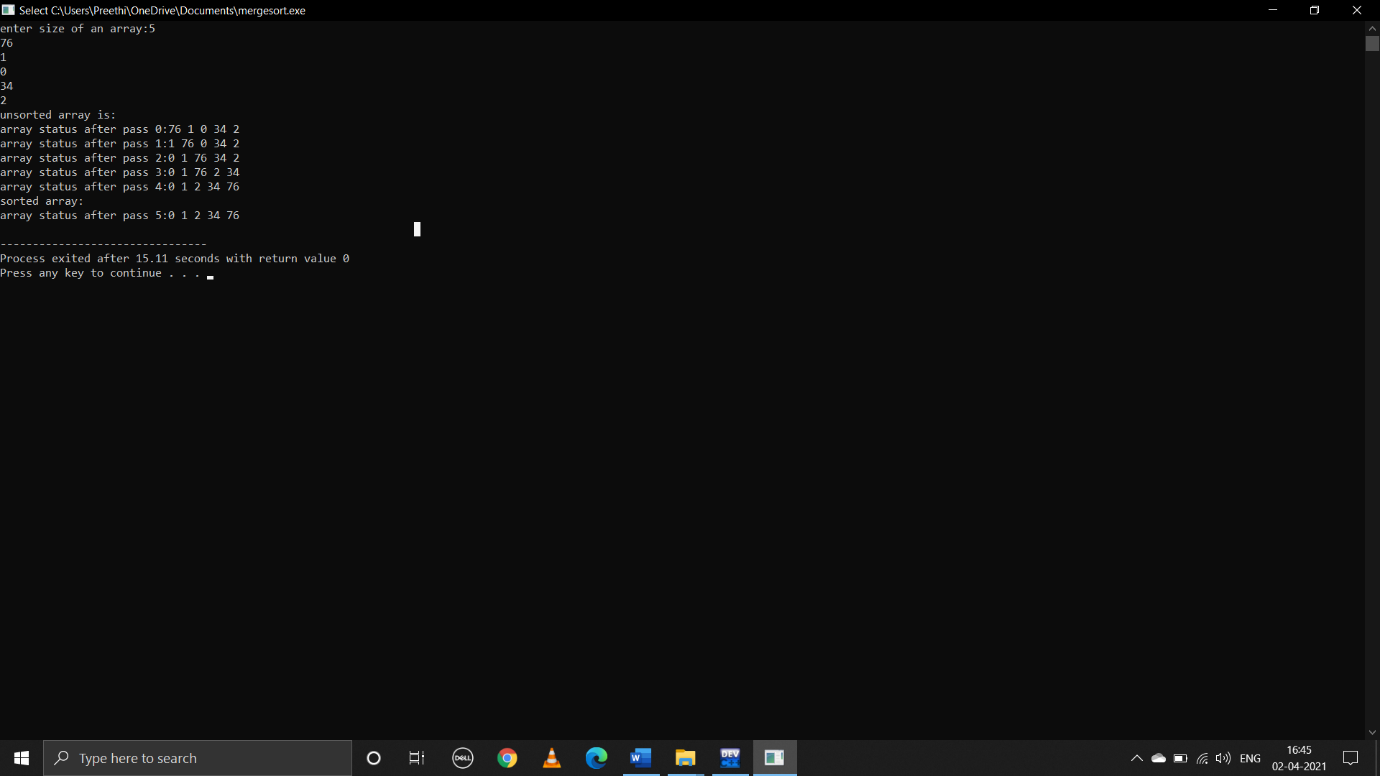
printarray(arr,n);

return 0;

}

**Output:**

****

****

**Done by:**

B.preethi

ID:B181356

Branch:cse

Class:AB-2,305