**PRACTICAL - 1**

1. Given an array of integers, find out the occurrences of a particular number.

Assume two scenarios:  1. Array is unsorted

                                      2. Array is sorted

**Exercise:**

1.1 Let say you are a cashier in PQR Bank, people come to you when they want to withdraw money from their respective account. People have to come with their ADHAR CARD Photocopy. Prakash came and withdrew money from his account, of course by submitting adhar card photocopy, but on the second day he felt that submitted adhar card photocopy was not his, but adhar card no. he knew, which is (89891245).

i) How will you find that particular adhar card? (Suppose total 15 people withdrew their money on that day). Assume there is no fix pattern in all adhar cards. )

ii) How will you find that particular adhar card? (Suppose total 20 people withdrew their money on that day). Assume there is a pattern in all adhar cards, all are in ascending order (89891235 To 89891254).

SOLUTION:

1. **LINEAR SEARCH(TIME COMPLEXITY:O(n)):**

#include<iostream>

using namespace std;

int main()

{

string a[15]= {"89845245","89896755","89456245","89834245","89891565","89891345","89141245",

"89821245","89891435","89891245","89895245","86891245","89491245","89823245",

"89891242"};

for(int i=0;i<15;i++)

{

if(a[i]=="89896755")

{

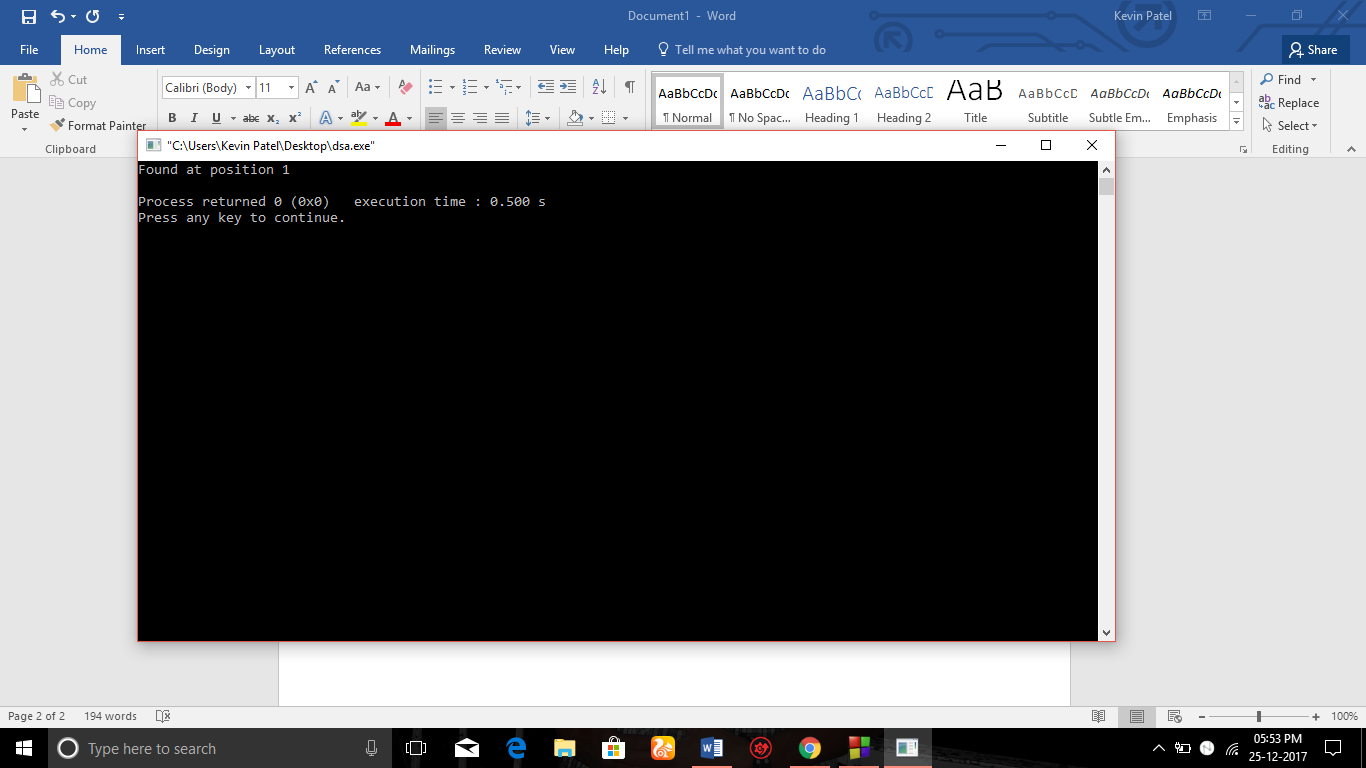
cout<<"Found at position "<<i<<endl;

**PRACTICAL - 1**

}

}

OUTPUT :



**PRACTICAL - 1**

**b.)BINARY SEARCH(TIME COMPLEXITY:O(log2n)):**

#include<iostream>

using namespace std;

int main()

{

long a[20],j=89891235,t=89891245;

for(int i=0;i<20;i++)

{

a[i]=j;

j++;

}

int l=0,u=20;

while(l<=u)

{

int midpoint=(l+u)/2;

if(t==a[midpoint])

{

cout<<"Found at position "<<midpoint<<endl;

break;

}

else if(t>a[midpoint])

{

l=midpoint;

}

else if(t<a[midpoint])

{

u=midpoint-1;

}

**PRACTICAL - 1**

}

if(l>u)

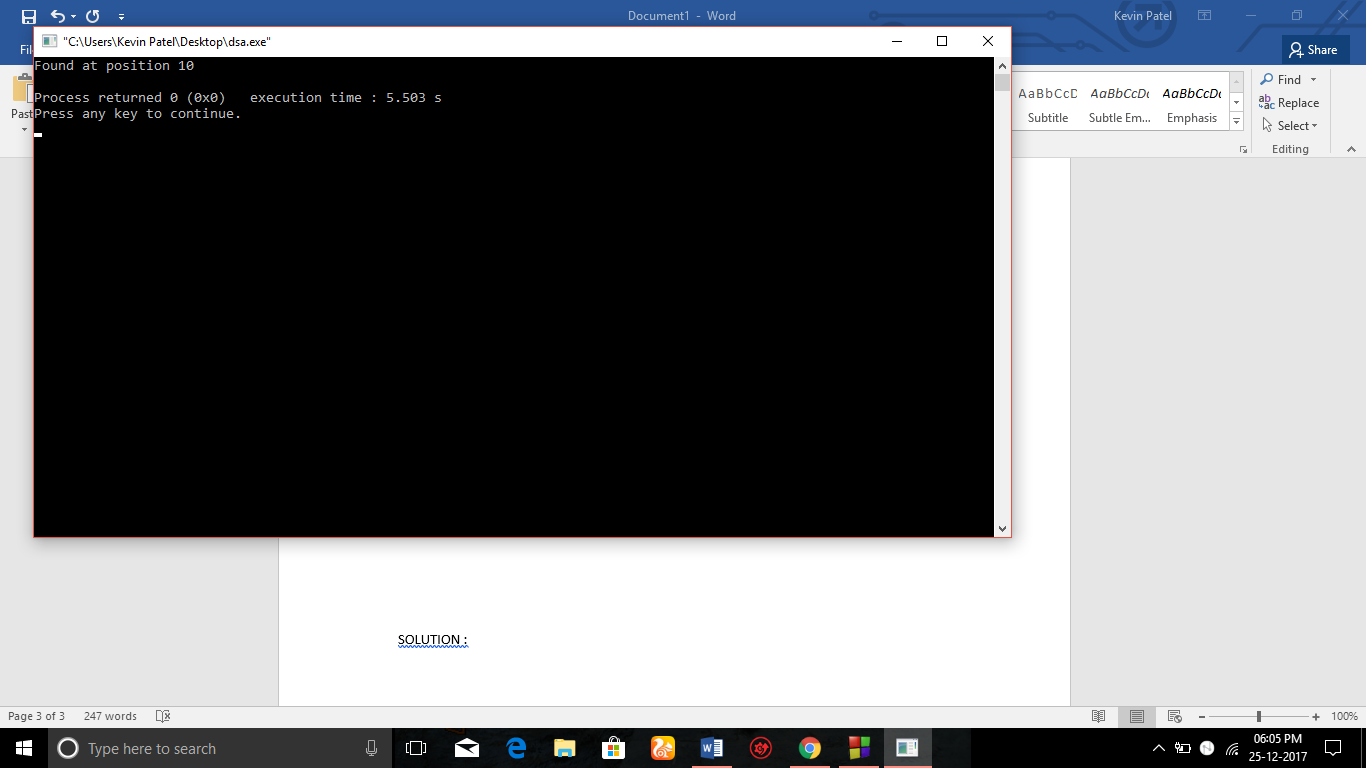
{

cout<<"THE NUMBER NOT FOUND"<<endl;

}

}

SOLUTION :



**PRACTICAL - 2**

1. Write a program to sort the following two lists using Selection and Insertion sort. Also,Count the number of comparisons and exchanges for both inputs.

Input1: 23,34,56,90,78,100,123,234

Input2: 34,54,12,10,67,45,55,88,10

**Exercise:**

2.1 Mark purchased Books from books store of standard 1 to 7. He purchased 4 books for each standard(for std.1 books are 1.1,1.2,1.3,1.4 and for std. 2 books are 2.1,2.2,2.3,2.4 and so on..). When he reached home, he opens the bag and sees that all the books got mixed. So, how he will sort all the books, according to the standards and their preference in that particular standard. (ex. : preference in std. 1 is 1.1<1.2<1.3)

2.2 Implement the function shift\_element() which takes as input the index of an array element that has been determined to be out of order. The function shifts the element towards the front of the array, repeatedly swapping the preceding element until the out-of-order element is in the proper location. Print the updated list.

SOLUTION :

1. SELECTION SORT .

#include<iostream>

using namespace std;

int main()

{

int i=0,j=0;

int a[8]={23,34,56,90,78,100,123,234},cmp=0;

for(i=0;i<8;i++)

{

int min=a[i];

int b;

for(j=i;j<8;j++)

{

**PRACTICAL - 2**

if(a[j]<=min)

{

min=a[j];

b=j;

cmp++;

}

}

int temp;

temp=a[i];

a[i]=a[b];

a[b]=temp;

}

for(i=0;i<8;i++)

{

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

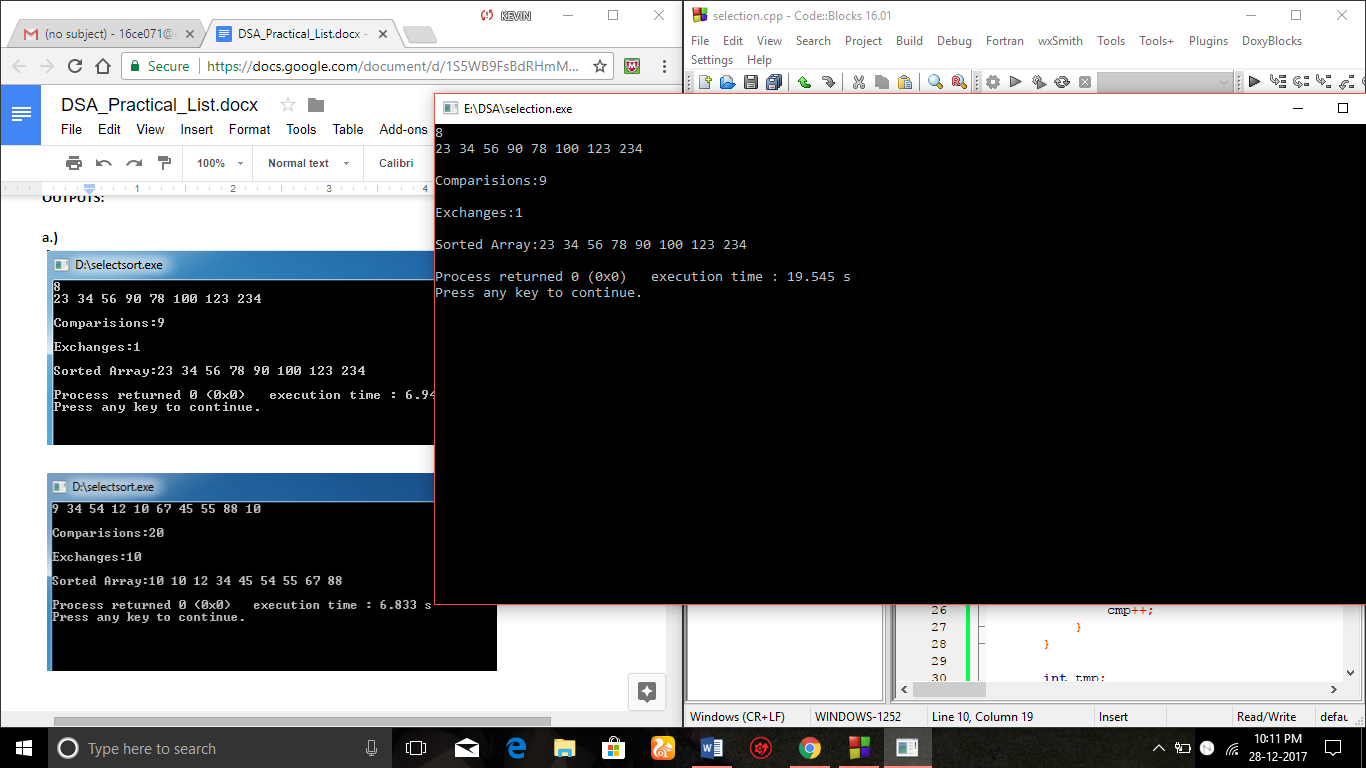
}

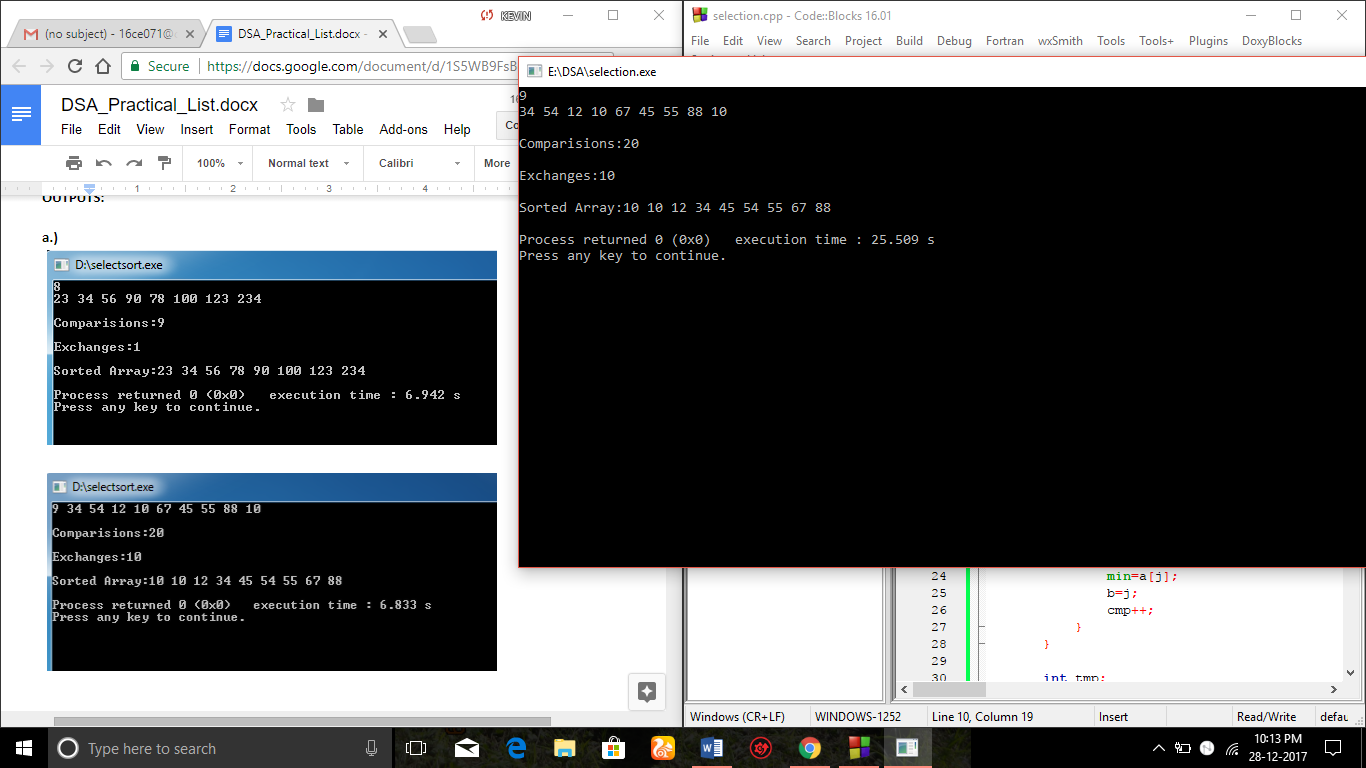
cout<<"\nCompared : "<<cmp;

}

**PRACTICAL - 2**

OUTPUT :





**PRACTICAL - 2**

1. INSERTION SORT

#include<iostream>

using namespace std;

int main()

{

int i,j,n;

cin>>n;

int a[n];

int k,cmp=0;

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

{

cin>>a[i];

}

cout<<endl;

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

{

k=a[j];

i=j-1;

while(i>=0 and a[i]>k)

{

a[i+1]=a[i];

i--;

cmp++;

}

a[i+1]=k;

}

cout<<"Sorted Array:";

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

**PRACTICAL - 2**

{

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

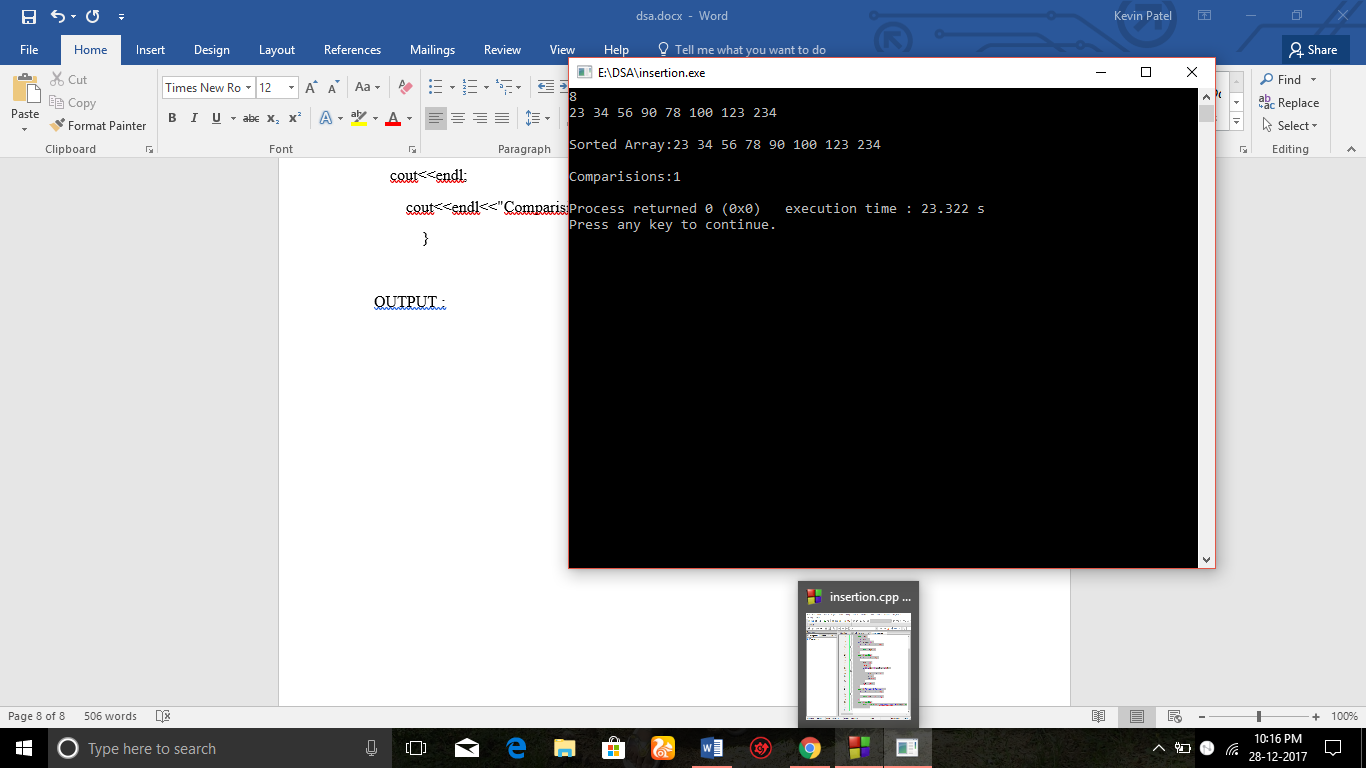
}

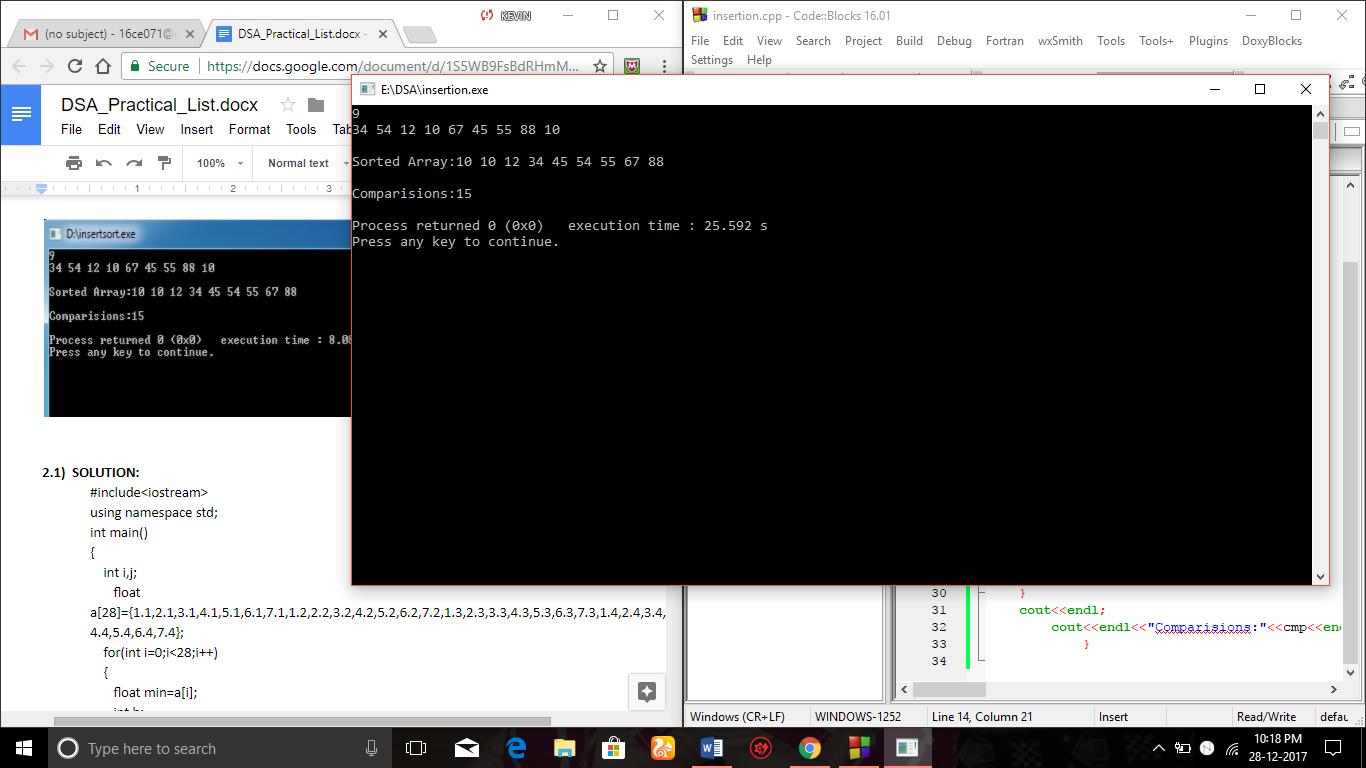
cout<<endl;

cout<<endl<<"Comparisions:"<<cmp<<endl;

}

OUTPUT :





**PRACTICAL - 2**

2.1)SOLUTION

#include<iostream>

using namespace std;

int main()

{

int i,j;

float a[28]={1.1,2.1,3.1,4.1,5.1,6.1,7.1,1.2,2.2,3.2,4.2,5.2,6.2,7.2,1.3,2.3,3.3,4.3,5.3,6.3,7.3,1.4,2.4,3.4,4.4,5.4,6.4,7.4};

for(int i=0;i<28;i++)

{

float min=a[i];

int b;

for(j=i;j<28;j++)

{

if(a[j]<=min)

{

min=a[j];

b=j;

}

}

float tmp;

tmp=a[i];

a[i]=a[b];

a[b]=tmp;

}

cout<<"After Sorting:"<<endl<<endl;

for(int i=0;i<28;i++)

**PRACTICAL - 2**

{

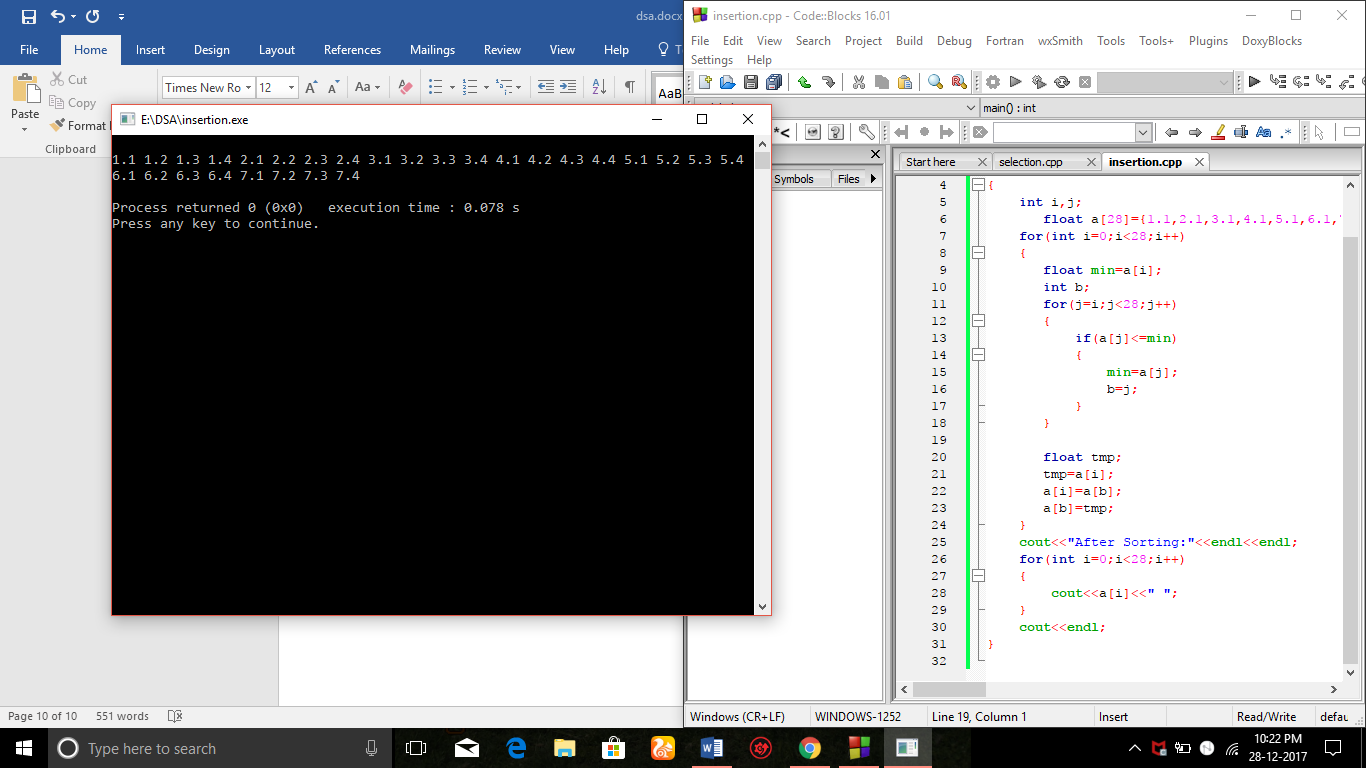
cout<<a[i]<<" ";

}

cout<<endl;

}

OUTPUT :

`

**PRACTICAL - 2**

2.2)SOLUTION

#include<iostream>

using namespace std;

void shift\_element(int i,int a[])

{

int j;

int temp=a[i];

for(j=i-1;j>=0;j--)

{

if(a[j]>temp)

{

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

}

else break;

}

a[j+1]=temp;

}

int main()

{

int n;

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

cin>>n;

int a[n];

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

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

{

cin>>a[i];

**PRACTICAL - 2**

}

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

{

shift\_element(i,a);

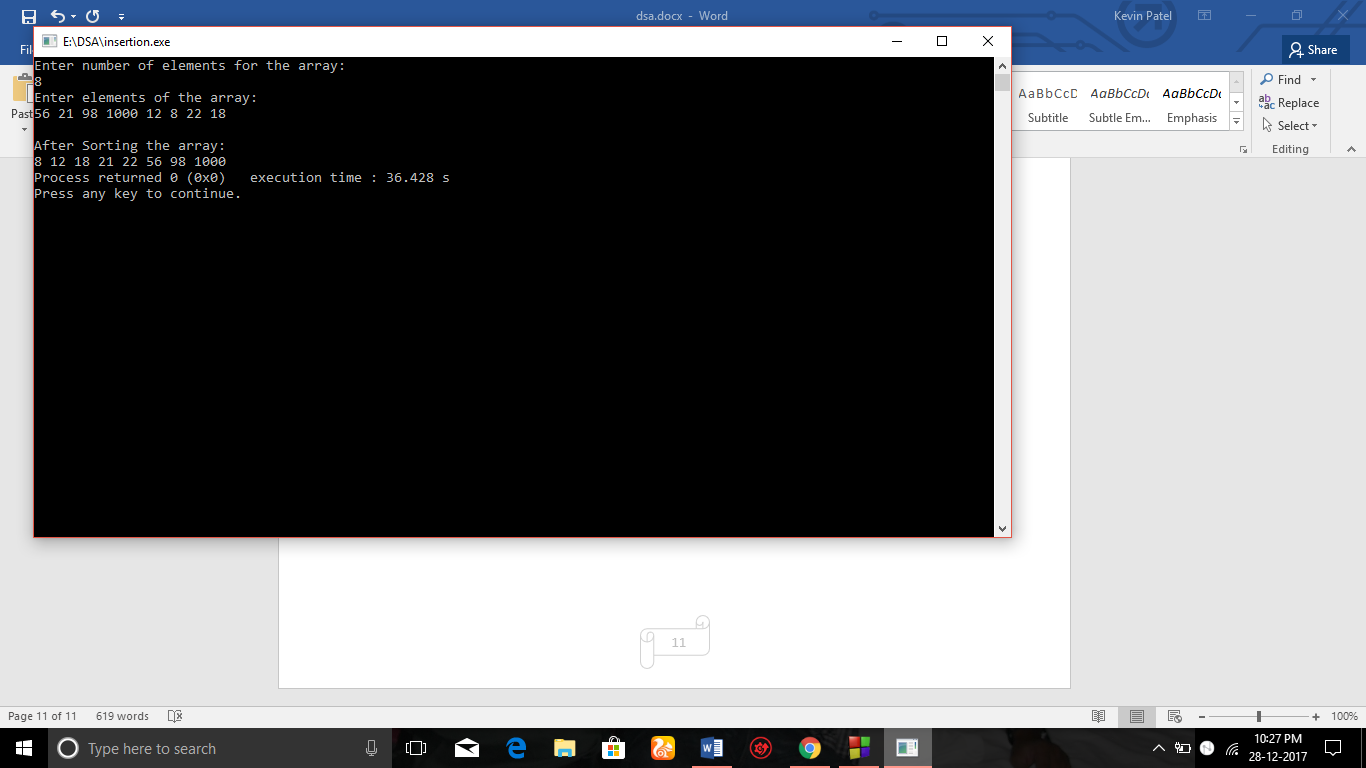
}

cout<<endl<<"After Sorting the array:"<<endl;

for(int i=0;i<n;i++) cout<<a[i]<<" ";

}

OUTPUT :



**PRACTICAL - 3**

3. Implement a menu driven program that performs following sorting algorithms.

3.1 QUICK SORT that arranges in ascending order.

3.2 MERGE SORT that arranges in descending order.

SOLUTION :

#include<iostream>

#include<stdlib.h>

using namespace std;

void quick\_sort(int[],int,int);

int partition(int[],int,int);

void quick\_sort(int a[],int l,int u)

{

int j;

if(l<u)

{

j=partition(a,l,u);

quick\_sort(a,l,j-1);

quick\_sort(a,j+1,u);

}

}

int partition(int a[],int l,int u)

{

int v,i,j,temp;

v=a[l];

i=l;

j=u+1;

**PRACTICAL - 3**

do

{

do

i++;

while(a[i]<v&&i<=u);

do

j--;

while(v<a[j]);

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}while(i<j);

a[l]=a[j];

a[j]=v;

return(j);

}

void merge(int A[], int l, int m, int r)

{

int i, j, k;

int n1 = m - l + 1;

**PRACTICAL - 3**

int n2 = r - m;

int L[n1], R[n2];

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

L[i] = A[l + i];

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

R[j] = A[m + 1+ j];

i = 0;

j = 0;

k = 0;

while (i < n1 && j < n2)

{

if (L[i] >= R[j])

{

A[k] = L[i];

i++;

}

else

{

A[k] = R[j];

j++;

}

k++;

}

while (i < n1)

{

**PRACTICAL - 3**

A[k] = L[i];

i++;

k++;

}

while (j < n2)

{

A[k] = R[j];

j++;

k++;

}

}

void mergeSort(int A[], int l, int r)

{

if (l < r)

{

int m = l+(r-l)/2;

mergeSort(A, l, m);

mergeSort(A, m+1, r);

merge(A, l, m, r);

}

}

int main()

{

int a[50],n,i;

cout<<"How many elements?";

cin>>n;

**PRACTICAL - 3**

cout<<"\nEnter array elements:";

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

cin>>a[i];

int enter;

while(1)

{

cout<<"\n1.Quick Sort \n2.Merge Sort \n3.Exit"<<endl;

cout<<"\nEnter your choice : ";

cin>>enter;

switch(enter)

{

case 1: quick\_sort(a,0,n-1);

cout<<"\nArray after sorting:";

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

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

cout<<endl<<"-----------------------------------------------"<<endl;

break;

case 2:mergeSort(a,0,n-1)

cout<<"Sorted array is : ";

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

{

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

}

cout<<endl<<"-----------------------------------------------"<<endl;

break;

case 3:cout<<"Thank you ";

**PRACTICAL - 3**

exit(0);

}

}

return 0;

}

OUTPUT :

