**Experiment No. 8**

**Title :** Implementation of Quick Sort algorithm

**Problem Statement :** Write a C++ program to arrange the given set of numbers in ascending order using Quick sort

**Algorithm:**

**Step 1:** Start

**Step 2:** Declare partition() function which divides the given array into equal or unequal halves

**Step 3:** Declare a function Quicksort() which takes array to be sorted and its size and performs following

* Choose the lowest index value has pivot
* Take two variables to point left and right of the list excluding pivot
* left points to the low index
* right points to the high
* while value at left is less than pivot move right
* while value at right is greater than pivot move left
* if both step 5 and step 6 does not match swap left and right
* if left ≥ right, the point where they met is new pivot

**Step 4:** Input the array to be sorted

**Step 5:** Call the Quicksort() function on the inputted array and display the sorted array

**Step 6:** Stop

**Program:**

#include <iostream>

using namespace std;

void quickSort(int[],int,int);

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

int main()

{

int a[50],n,i;

cout<<"How many elements to be sorted? :";

cin>>n;

cout<<"\nEnter the elements :";

for(i=0;i<n;i++) //To read the array elements

cin>>a[i];

quickSort(a,0,n-1);

cout<<"\nArray after sorting : ";

for(i=0;i<n;i++) //To print the sorted array

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

return 0;

}

void quickSort(int a[],int low,int up)

{

int j,i;

if(low<up)

{

j=partition(a,low,up);

// The following 3 blocks of code is to observe the partitioning

// and intermediate results

cout<<"\n";

for(i=low;i<=j-1;i++) //To print the array

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

cout<<"\n";

cout<<"\n";

for(i=j+1;i<=up;i++) //To print the array

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

cout<<"\n";

cout<<"\n";

for(i=0;i<=7;i++) //To print the array

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

cout<<"\n";

quickSort(a,low,j-1);

quickSort(a,j+1,up);

}

}

int partition(int a[],int low,int up)

{

int piv,i,j,temp;

piv=a[low]; //piv is the element whose final position is sought

i=low;

j=up+1;

cout<< "\n piv="<<piv<<"\n";

do

{

do

i++; // move up the array

while(a[i]< piv && i<=up);

do

j--; // move dpwn the array

while(piv<a[j]);

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}while (i<j);

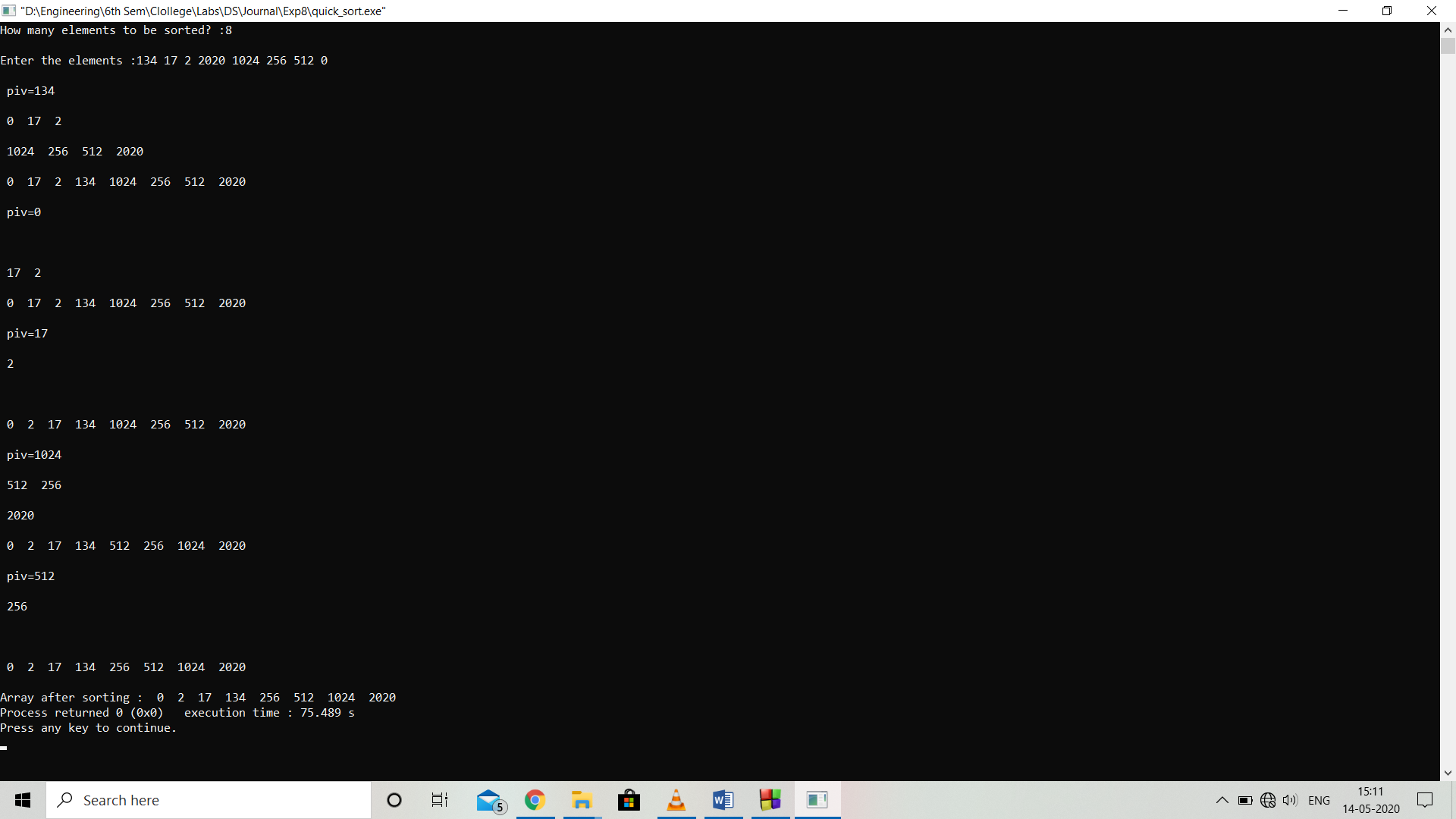
a[low]=a[j];

a[j]=piv;

return(j);

}

**Output :**

****

**Analysis :**

Quick sort is better than bubble sort as the iterations done to sort an array of ‘n’ elements is ‘n\*log(n)’ in most of the times which are lesser then ‘n\*n’ which are of bubble sort

**Limitation:**

If the pivot value picked is largest value in the given array then the quick sort will perform ‘n\*n’ iterations instead of ‘n\*log(n)’ for an array of ‘n’ elements which slows down the sorting process.