**Objective -**

Write a program to implement Quick sort.

**Description-**

**Quick sort –**

* QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

1. Always pick first element as pivot.
2. Always pick last element as pivot (implemented below)
3. Pick a random element as pivot.
4. Pick median as pivot.

* The key process in quickSort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time
* Quick Sort is also based on the concept of **Divide and Conquer**, just like merge sort. But in quick sort all the heavy lifting(major work) is done while **dividing** the array into subarrays, while in case of merge sort, all the real work happens during **merging** the subarrays. In case of quick sort, the combine step does absolutely nothing.
* It is also called **partition-exchange sort**. This algorithm divides the list into three main parts:

1. Elements less than the **Pivot** element
2. Pivot element(Central element)
3. Elements greater than the pivot element

* **Pivot** element can be any element from the array, it can be the first element, the last element or any random element. In this tutorial, we will take the rightmost element or the last element as **pivot**.

Program –

#include<stdio.h>

#include<conio.h> //header files

int partition(int[],int,int); //function prototype

/\* creating partitions and then recursively sorting the partions \*/

void quicksort(int x[],int lb,int ub)

{

int p;

if(lb < ub) // if statement

{

p = partition(x,lb,ub);

quicksort(x,lb,p-1);

quicksort(x,p+1,ub);

}

}

/\* sorting the elements of the partition in ascending order \*/

int partition(int x[],int lb,int ub)

{

int val=x[lb],down = lb+1,up = ub,t;

while(down <= up) // while loop

{

while(down <= up && x[down] < val)

down++; // increment

while(x[up] > val)

up--;

if(down < up)

{ //swap

t = x[down];

x[down] = x[up];

x[up] = t;

}

}

//swap the pivot element

x[lb] = x[up];

x[up] = val;

return up; // return position of pivot element

}

/\* starting point of the program \*/

int main(void)

{

int x[20],i,n;

clrscr();

printf("\n\tEnter the number of elements=");

scanf("%d", &n);

printf("\n\tEnter the elements\n\t");

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

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

quicksort(x,0,n-1); // function calling

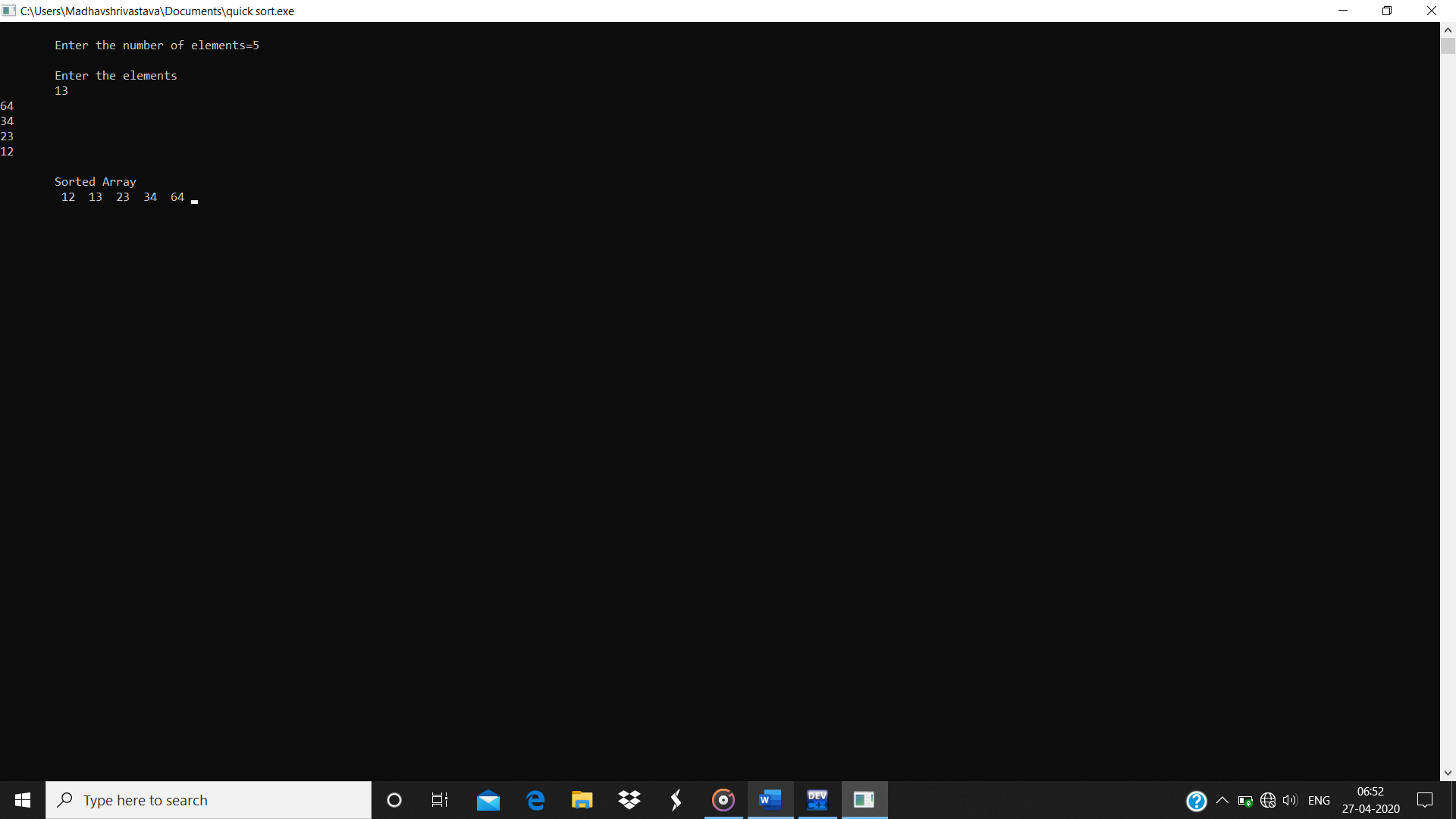
printf("\n\tSorted Array\n\t");

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

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

getch();

}

Output –