Ex. No:11.B.3 **QUICK SORT**

AIM:

To write a C program to implement the concept of Quick sort.

DESCRIPTION:

Quick Sort, as the name suggests, sorts any list very quickly. Quick sort is not stable search, but it is very fast and requires very less additional space. It is based on the rule of Divide and Conquer (also called partition-exchange sort). This algorithm divides the list into three main parts:

- Elements less than the Pivot element
- Pivot element
- Elements greater than the pivot element

ALGORITHM:

1: Start.

2: Choose any element of the array to be the pivot.

3: Divide all other elements (except the pivot) into two partitions.

- All elements less than the pivot must be in the first partition.
- All elements greater than the pivot must be in the second partition.

4: Use recursion to sort both partitions.

5: Join the first sorted partition, the pivot, and the second sorted partition.

6: Stop

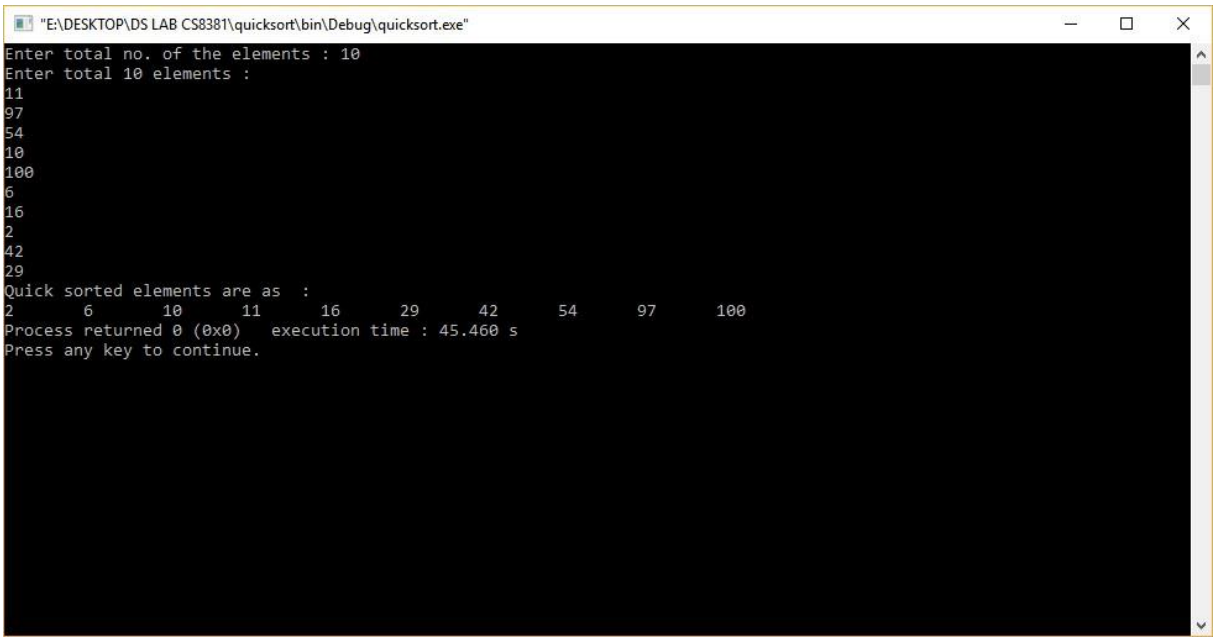
PROGRAM:

```
#include<stdio.h> #include<conio.h>

void qsort(int arr[20], int fst, int
```

```
last); void main(){ int arr[30]; int
i,size;
 printf("Enter total no. of the elements
: "); scanf("%d",&size); printf("Enter
total %d elements : \n",size); for(i=0;
i<size; i++) scanf("%d",&arr[i]);
qsort(arr,0,size-1); printf("Quick
sorted elements are as  : \n"); for(i=0;
i<size; i++) printf("%d\t",arr[i]);
getch();} void qsort(int arr[20], int
fst, int last){ int i,j,pivot,tmp;
if(fst<last){ pivot=fst; i=fst; j=last;
while(i<j){ while(arr[i]<=arr[pivot] &&
i<last) i++; while(arr[j]>arr[pivot]) j-
-; if(i<j){ tmp=arr[i]; arr[i]=arr[j];
arr[j]=tmp;}} tmp=arr[pivot];
arr[pivot]=arr[j];
  arr[j]=tmp;
qsort(arr,fst,j
-1);
qsort(arr,j+1,l
ast);
} }
```

OUTPUT



RESULT:

Thus the C program to implement the concept of Quick sort.