Program

```c
#include <stdio.h>

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int arr[], int first, int last)
{
    int pivot = arr[last];
    int i = first - 1;

    for (int j = first; j < last; j++)
    {
        if (arr[j] <= pivot)
        {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }

    swap(&arr[i + 1], &arr[last]);
    return i + 1;
}

void quickSort(int arr[], int first, int last)
{
    if (first < last)
    {
        int pivotIndex = partition(arr, first, last);
        quickSort(arr, first, pivotIndex - 1);
        quickSort(arr, pivotIndex + 1, last);
    }
}

int main()
{
    int arr[50], n, i;

    printf("Enter size of the array: ");
    scanf("%d", &n);

    printf("Enter %d elements: ", n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }

    quickSort(arr, 0, n - 1);

    printf("\nSorted array: ");
    for (i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Experiment No:23
Date:

QUICK SORT

Aim:

To sort a given array using quick sort

Algorithm:

1. Start
2. Create a swap function

```
temp=*a
*a=*b
*b=temp
```

3. Create function Partition

```
pivot = arr[last]
i = first- 1
for j from first to last - 1:
  if arr[j]<= pivot
      i = i + 1
      Swap(arr[i], arr[j])
Swap(arr[i + 1], arr[last])
return i + 1
```

4. Create a function QuickSort

```
if first < last:
   pivotIndex = Partition(arr, first,last )
   QuickSort(arr, first, pivotIndex - 1)
   QuickSort(arr, pivotIndex + 1, last)
```

5. Create a main function

```
Print enter the size of the array
Print enter the elements of the array
for i=0 to n-1
   call quicksort function
   print arr[i]
```

6. Stop

Output

```
Enter size of the array:5
Enter 5 elements: 21 46 78 31 65
Sorted array:21 31 46 65 78
```

Result:

Program has been executed successfully and obtained the output