PRACTICAL-03

//WAP to implement quick sort:

Practical-03

//Algorithm:-

Quick Sort(A,p,r)

1. If p<r
2. q <-- Partition (A,p,r)
3. Quick Sort(A,p,q-1)
4. Quick Sort(A,q+1,r)
5. Partition the array

PARTITION(A,p,r)

1. x <-- A[r]
2. i <--p-1
3. for j <-- to r-1
4. do if A[j] <= x
5. then i <-- i+1
6. exchange A[i] <--> A[j]
7. exchange A[i+1] <--> A[r]
8. return i+1

//Code

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

```
void printArray(int A[], int n)
{
   for (int i = 0; i < n; i++)
   {
      printf("%d ", A[i]);
   }
   printf("\n");
}

int partition(int A[], int low, int high)
{
   int pivot = A[low];
   int i = low + 1;
   int j = high;
   int temp;

   do
   {
      while (A[i] <= pivot)
      {
         i++;
      }

      while (A[j] > pivot)
      {
         j--;
      }
```

```
        if (i < j)
        {
            temp = A[i];
            A[i] = A[j];
            A[j] = temp;
        }
    } while (i < j);

    // Swap A[low] and A[j]
    temp = A[low];
    A[low] = A[j];
    A[j] = temp;
    return j;
}

void quickSort(int A[], int low, int high)
{
    int partitionIndex; // Index of pivot after partition

    if (low < high)
    {
        partitionIndex = partition(A, low, high);
        quickSort(A, low, partitionIndex - 1);  // sort left subarray
        quickSort(A, partitionIndex + 1, high); // sort right subarray
    }
}

int main()
{
```

```
    //int A[] = {3, 5, 2, 13, 12, 3, 2, 13, 45};
    int A[] = {9, 4, 4, 8, 7, 5, 6};
    // 3, 5, 2, 13, 12, 3, 2, 13, 45
    // 3, 2, 2, 13i, 12, 3j, 5, 13, 45
    // 3, 2, 2, 3j, 12i, 13, 5, 13, 45 --> first call to partition returns 3
    int n = 9;
    n =7;
    printArray(A, n);
    quickSort(A, 0, n - 1);
    printArray(A, n);
    return 0;
}
```

output