```c
//Quick sort algorithm is implememnted by using 2 concepts:
/*1)Divide and conquer
2)Partition method*/

//Choosing a pivot for every partition we assume
//Every time you call a partition, the pivot element gets it

#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)
        {
```

```
40:             temp = A[i];
41:             A[i] = A[j];
42:             A[j] = temp;
43:         }
44:     } while (i < j);
45:
46:     // Swap A[low] and A[j]
47:     temp = A[low];
48:     A[low] = A[j];
49:     A[j] = temp;
50:     return j;
51: }
52:
53: void quickSort(int A[], int low, int high)
54: {
55:     int partitionIndex; // Index of pivot after partition
56:
57:     if (low < high)
58:     {
59:         partitionIndex = partition(A, low, high);
60:         quickSort(A, low, partitionIndex - 1);  // sort left subarr
61:         quickSort(A, partitionIndex + 1, high); // sort right subar
62:     }
63: }
64:
65: int main()
66: {
67:     //int A[] = {3, 5, 2, 13, 12, 3, 2, 13, 45};
68:     int A[] = {9, 4, 4, 8, 7, 5, 6};
69:     // 3, 5, 2, 13, 12, 3, 2, 13, 45
70:     // 3, 2, 2, 13i, 12, 3j, 5, 13, 45
71:     // 3, 2, 2, 3j, 12i, 13, 5, 13, 45 --> first call to partiti
72:     int n = 9;
73:     n =7;
74:     printArray(A, n);
75:     quickSort(A, 0, n - 1);
76:     printArray(A, n);
77:     return 0;
78: }
```

79:
80:
81: