```
1: #include <stdio.h>
 2:
 3: void printArray(int *A, int n)
4: {
 5:
         for (int i = 0; i < n; i++)</pre>
 6:
             printf("%d ", A[i]);
 7:
 8:
         printf("\n");
 9:
10: }
11:
12: int partition(int A[], int low, int high)
13: {
14:
         int pivot = A[low];
         int i=low+1;
15:
         int j=high;
16:
17:
         int temp;
18:
19:
         do
20:
         {
21:
               for(;i<=high && A[i]<pivot;i++){</pre>
22:
23:
             }
24:
25:
26:
27:
28:
29:
               for(; j>=low && A[j]>pivot; j--){
30:
             }
31:
32:
             if (i < j)
33:
34:
35:
                  temp = A[i];
                 A[i] = A[j];
36:
37:
                  A[j] = temp;
38:
         } while (i < j);</pre>
39:
```

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