

## Question01.cpp

```
1 // <--- DSA LAB 8 --->
2 // <--- Q1 --->
3
4 // Write a program to implement a recursive version of merge-sort. Run it for some sample
5 // data.
6
7 #include<iostream>
8 using namespace std;
9
10 void merge(int *arr, int s, int e) {
11
12     int mid = (s+e)/2;
13
14     int len1 = mid - s + 1;
15     int len2 = e - mid;
16
17     int *first = new int[len1];
18     int *second = new int[len2];
19
20     //copy values
21     int mainArrayIndex = s;
22     for(int i=0; i<len1; i++) {
23         first[i] = arr[mainArrayIndex++];
24     }
25
26     mainArrayIndex = mid+1;
27     for(int i=0; i<len2; i++) {
28         second[i] = arr[mainArrayIndex++];
29     }
30
31     //merge 2 sorted arrays
32     int index1 = 0;
33     int index2 = 0;
34     mainArrayIndex = s;
35
36     while(index1 < len1 && index2 < len2) {
37         if(first[index1] < second[index2]) {
38             arr[mainArrayIndex++] = first[index1++];
39         }
40         else{
41             arr[mainArrayIndex++] = second[index2++];
42         }
43     }
44
45     while(index1 < len1) {
46         arr[mainArrayIndex++] = first[index1++];
47     }
48
49     while(index2 < len2 ) {
50         arr[mainArrayIndex++] = second[index2++];
51     }
52
53     delete []first;
```

```
54     delete []second;
55
56 }
57
58 void mergeSort(int *arr, int s, int e) {
59
60     //base case
61     if(s >= e) {
62         return;
63     }
64
65     int mid = (s+e)/2;
66
67     //left part sort karna h
68     mergeSort(arr, s, mid);
69
70     //right part sort karna h
71     mergeSort(arr, mid+1, e);
72
73     //merge
74     merge(arr, s, e);
75
76 }
77
78 int main() {
79
80     int arr[15] = {3,7,0,1,5,8000,3,2000,34,66,870,230,128,120,122};
81     int n = 15;
82
83     mergeSort(arr, 0, n-1);
84
85     for(int i=0;i<n;i++){
86         cout << arr[i] << " ";
87     } cout << endl;
88
89     return 0;
90 }
```

## Question02.cpp

```
1 // <--- DSA LAB 8 --->
2 // <--- Q2 --->
3
4 // Write a program to implement a recursive version of quicksort. Run it for some sample
5 // data.
6
7 #include<iostream>
8 using namespace std;
9
10
11 int partition( int arr[], int s, int e) {
12
13     int pivot = arr[s];
14
15     int cnt = 0;
16     for(int i = s+1; i<=e; i++) {
17         if(arr[i] <=pivot) {
18             cnt++;
19         }
20     }
21
22     //place pivot at right position
23     int pivotIndex = s + cnt;
24     swap(arr[pivotIndex], arr[s]);
25
26     //left and right wala part smbhal lete h
27     int i = s, j = e;
28
29     while(i < pivotIndex && j > pivotIndex) {
30
31         while(arr[i] <= pivot)
32         {
33             i++;
34         }
35
36         while(arr[j] > pivot) {
37             j--;
38         }
39
40         if(i < pivotIndex && j > pivotIndex) {
41             swap(arr[i++], arr[j--]);
42         }
43
44     }
45
46     return pivotIndex;
47 }
48
49 void quickSort(int arr[], int s, int e) {
50
51     //base case
52     if(s >= e)
```

```
54         return ;
55
56         //partitioon karenge
57         int p = partition(arr, s, e);
58
59         //left part sort karo
60         quickSort(arr, s, p-1);
61
62         //right wala part sort karo
63         quickSort(arr, p+1, e);
64
65     }
66
67     int main() {
68
69         int arr[10] = {2,4,1,6,1000,8,5,0,8,10};
70         int n = 10;
71
72         quickSort(arr, 0, n-1);
73
74         for(int i=0; i<n; i++)
75         {
76             cout << arr[i] << " ";
77         } cout << endl;
78
79
80         return 0;
81     }
```