

Name: Mithlesh Yeole

Roll no.: B3-B3-59

## PRACTICAL 8

QUICK SORT implementation:

Code:

```

C quicksort.c X C mergesort.c
C quicksort.c > main()
1  #include<stdio.h>
2
3  void swap(int *a, int *b){
4      int temp = *a;
5      *a = *b;
6      *b = temp;
7  }
8  int partition_first(int a[], int lb, int ub){
9      int pivot = a[lb];
10     int start = lb;
11     int end = ub;
12     while (start < end) {
13         while (a[start] <= pivot)
14             start++;
15         while (a[end] > pivot)
16             end--;
17         if (start < end) {
18             swap(&a[start], &a[end]);
19         }
20     }
21     swap(&a[lb], &a[end]);
22     return end;
23 }
24
25 void quicksort_first(int a[], int lb, int ub){
26     if(lb < ub) {
27         int loc = partition_first(a, lb, ub);
28         quicksort_first(a, lb, loc - 1);
29         quicksort_first(a, loc + 1, ub);
30     }
31 }
32 int main(){
33     int size;
34     printf("Enter the size of the array: ");
35     scanf("%d", &size);
36     int a[size];
37     printf("Enter the elements of the array: ");

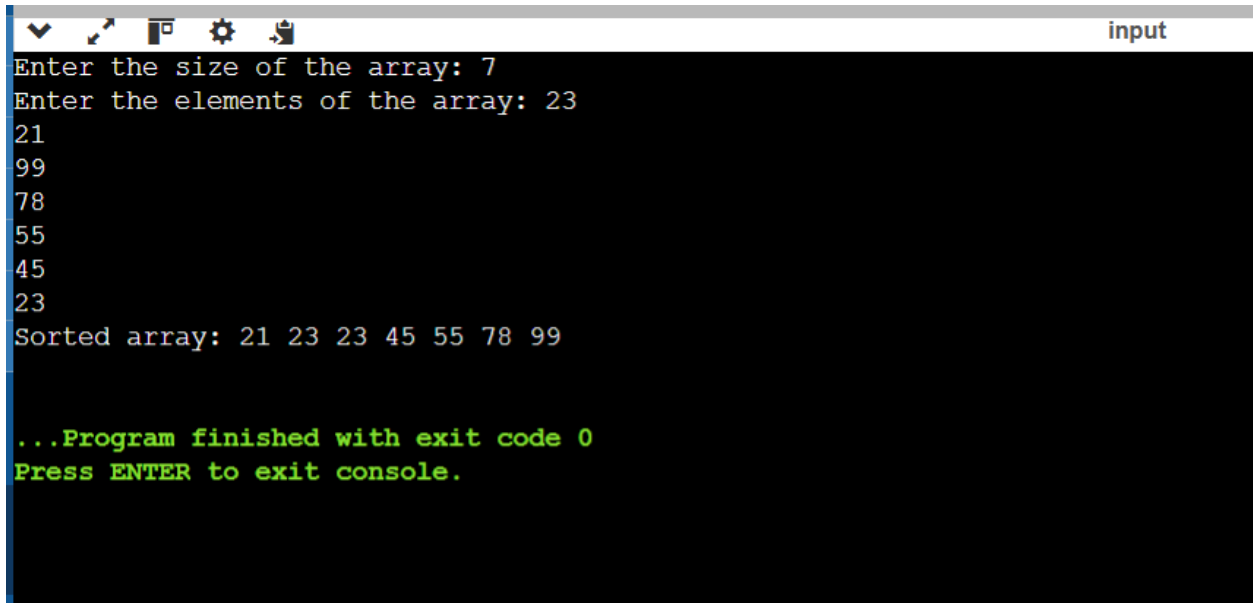
```

```

C quicksort.c X C mergesort.c
C quicksort.c > partition_first(int [], int, int)
31 }
32 int main(){
33     int size;
34     printf("Enter the size of the array: ");
35     scanf("%d", &size);
36     int a[size];
37     printf("Enter the elements of the array: ");
38     for(int i = 0; i < size; i++){
39         scanf("%d", &a[i]);
40     }
41     quicksort_first(a, 0, size - 1);
42     printf("Sorted array: ");
43     for(int i = 0; i < size; i++){
44         printf("%d ", a[i]);
45     }
46     printf("\n");
47     return 0;
48 }

```

Output:



```
Enter the size of the array: 7
Enter the elements of the array: 23
21
99
78
55
45
23
Sorted array: 21 23 23 45 55 78 99

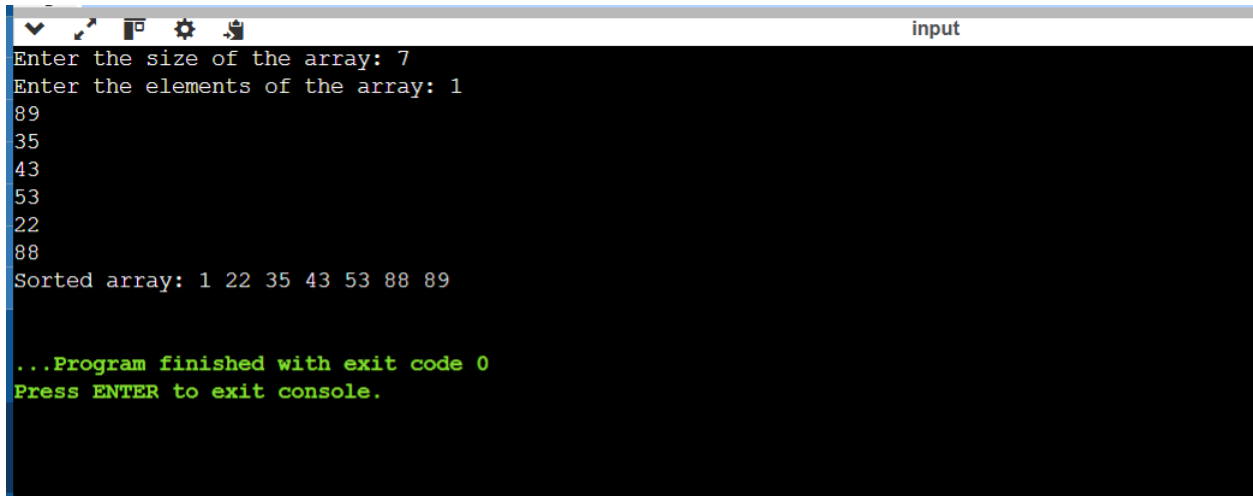
...Program finished with exit code 0
Press ENTER to exit console.
```

## MERGE SORT implementation:

```
Welcome | C quicksort.c | C mergesort.c X
C mergesort.c > merge(int [], int, int, int)
1 #include<stdio.h>
2 #define size 100
3
4 void merge(int a[], int beg, int mid, int end){
5     int i = beg, j = mid + 1, index = beg;
6     int temp[size], k;
7
8     while(i <= mid && j <= end){
9         if(a[i] < a[j]){
10             temp[index] = a[i];
11             i++;
12         } else {
13             temp[index] = a[j];
14             j++;
15         }
16         index++;
17     }
18
19     while(i <= mid){
20         temp[index] = a[i];
21         i++;
22         index++;
23     }
24
25     while(j <= end){
26         temp[index] = a[j];
27         j++;
28         index++;
29     }
30
31     for(k = beg; k < index; k++){
32         a[k] = temp[k];
33     }
34 }
35
36 void merge_sort(int a[], int beg, int end){
37     int mid;
38
39     if(beg < end){
40         mid = (beg + end) / 2;
41         merge_sort(a, beg, mid);
42         merge_sort(a, mid + 1, end);
43         merge(a, beg, mid, end);
44     }
45 }
46
47 int main(){
48     int arr[size], i, n;
49     printf("Enter the number of elements in the array: ");
50     scanf("%d", &n);
51
52     printf("Enter %d elements in the array:\n", n);
53     for(i = 0; i < n; i++){
54         scanf("%d", &arr[i]);
55     }
56
57     merge_sort(arr, 0, n - 1);
58
59     printf("Sorted array is:\n");
60     for(i = 0; i < n; i++){
61         printf("%d ", arr[i]);
62     }
63     printf("\n");
64
65     return 0;
66 }
```

```
Welcome | C quicksort.c | C mergesort.c X
C mergesort.c > merge(int [], int, int, int)
36 void merge_sort(int a[], int beg, int end){
37     int mid;
38     if(beg < end){
39         mid = (beg + end) / 2;
40         merge_sort(a, beg, mid);
41         merge_sort(a, mid + 1, end);
42         merge(a, beg, mid, end);
43     }
44 }
45
46 int main(){
47     int arr[size], i, n;
48     printf("Enter the number of elements in the array: ");
49     scanf("%d", &n);
50
51     printf("Enter %d elements in the array:\n", n);
52     for(i = 0; i < n; i++){
53         scanf("%d", &arr[i]);
54     }
55
56     merge_sort(arr, 0, n - 1);
57
58     printf("Sorted array is:\n");
59     for(i = 0; i < n; i++){
60         printf("%d ", arr[i]);
61     }
62     printf("\n");
63
64     return 0;
65 }
66
```

Output:



```
Enter the size of the array: 7
Enter the elements of the array: 1
89
35
43
53
22
88
Sorted array: 1 22 35 43 53 88 89

...Program finished with exit code 0
Press ENTER to exit console.
```