

Name: Devansh Upadhyay

Roll No. 53

Program for Bubble Sort:

```
DAA > Pract 1 > C bubble.c > bubble_sort(int [], int)
1  #include <stdio.h>
2  void bubble_sort(int[],int);
3  int main(){
4      int a[30],n,i;
5      printf("\nEnter no of elements: ");
6      scanf("%d",&n);
7      printf("\nEnter array Elements: ");
8      for(i=0;i<n;i++){
9          scanf("%d",&a[i]);
10     }
11     bubble_sort(a,n);
12     printf("\nSorted Array is: ");
13     for ( i = 0; i < n; i++)
14     {
15         printf("%d",a[i]);
16         putchar(' ');
17     }
18 }
19 void bubble_sort(int a[],int n){
20     int i,j,temp,flag;
21     flag=1;
22     for ( i = 1; i < n && flag==1; i++)
23     {
24         flag=0;
25         for(j=0; j<n-i;j++){
26             if(a[j]>a[j+1]){
27                 flag=1;
28                 temp=a[j];
29                 a[j]=a[j+1];
30                 a[j+1]=temp;
31             }
26 }
```

Output for Bubble Sort:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS D:\5th-Sem-Practicals\DAA\Pract 1> gcc bubble.c
PS D:\5th-Sem-Practicals\DAA\Pract 1> ./a.exe
```

Enter no of elements: 6

Enter array Elements: 55 44 33 11 22 66

Sorted Array is: 11 22 33 44 55 66

```
PS D:\5th-Sem-Practicals\DAA\Pract 1> █
```

Program for Merge Sort:

```
DAA > Pract 1 > C merge.c > ...
1  #include <stdio.h>
2  void merge(int arr[], int l, int m, int r)
3  {
4      int i, j, k;
5      int n1 = m - 1 + 1;
6      int n2 = r - m;
7      int L[n1], R[n2];
8      for (i = 0; i < n1; i++)
9          L[i] = arr[l + i];
10     for (j = 0; j < n2; j++)
11         R[j] = arr[m + 1 + j];
12     i = 0;
13     j = 0;
14     k = l;
15     while (i < n1 && j < n2) {
16         if (L[i] <= R[j]) {
17             arr[k] = L[i];
18             i++;
19         }
20         else {
21             arr[k] = R[j];
22             j++;
23         }
24         k++;
25     }
26     while (i < n1) {
27         arr[k] = L[i];
28         i++;
29         k++;
30     }
31     while (j < n2) {
32         arr[k] = R[j];
33         j++;
34         k++;
35     }
36 }
37 void mergeSort(int arr[], int l, int r)
38 {
39     if (l < r) {
40         int m = l + (r - l) / 2;
41         mergeSort(arr, l, m);
42         mergeSort(arr, m + 1, r);
43         merge(arr, l, m, r);
44     }
45 }
46 void printArray(int A[], int size)
47 {
48     int i;
49     for (i = 0; i < size; i++)
50         printf("%d ", A[i]);
51     printf("\n");
52 }
53 int main()
54 {
55     int arr[] = { 12, 11, 13, 5, 6, 7 };
56     int arr_size = sizeof(arr) / sizeof(arr[0]);
57     printf("Given array is \n");
58     printArray(arr, arr_size);
59     mergeSort(arr, 0, arr_size - 1);
60     printf("\nSorted array is \n");
61     printArray(arr, arr_size);
62     return 0;
63 }
```

Output for Merge Sort:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS D:\5th-Sem-Practicals\DAA\Pract 1> gcc merge.c
```

```
PS D:\5th-Sem-Practicals\DAA\Pract 1> ./a.exe
```

```
Given array is
```

```
12 11 13 5 6 7
```

```
Sorted array is
```

```
5 6 7 11 12 13
```

```
PS D:\5th-Sem-Practicals\DAA\Pract 1> █
```

Program for Quick Sort:

```
DAA > Pract 1 > C quick.c > swap(int *, int *)
1  #include <stdio.h>
2  void swap(int *a, int *b) {
3      int t = *a;
4      *a = *b;
5      *b = t;
6  }
7  int partition(int array[], int low, int high) {
8      int pivot = array[high];
9      int i = (low - 1);
10     for (int j = low; j < high; j++) {
11         if (array[j] <= pivot) {
12             i++;
13             swap(&array[i], &array[j]);
14         }
15     }
16     swap(&array[i + 1], &array[high]);
17     return (i + 1);
18 }
19 void quickSort(int array[], int low, int high) {
20     if (low < high) {
21         int pi = partition(array, low, high);
22         quickSort(array, low, pi - 1);
23         quickSort(array, pi + 1, high);
24     }
25 }
26 void printArray(int array[], int size) {
27     for (int i = 0; i < size; ++i) {
28         printf("%d ", array[i]);
29     }
30     printf("\n");
31 }
32 int main() {
33     int data[] = {8, 7, 2, 1, 0, 9, 6};
34     int n = sizeof(data) / sizeof(data[0]);
35     printf("Unsorted Array\n");
36     printArray(data, n);
37     quickSort(data, 0, n - 1);
38     printf("Sorted array: \n");
39     printArray(data, n);
40 }
```

Output for Quick Sort:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS D:\5th-Sem-Practicals\DAA\Pract 1> gcc quick.c
PS D:\5th-Sem-Practicals\DAA\Pract 1> ./a.exe
Unsorted Array
8 7 2 1 0 9 6
Sorted array:
0 1 2 6 7 8 9
PS D:\5th-Sem-Practicals\DAA\Pract 1> █
```