Program for Bubble Sort:

DAA > Pract 1 > C bubble.c > bubble_sort(int [], int)

```c
#include <stdio.h>
void bubble_sort(int[],int);
int main(){
    int a[30],n,i;
    printf("\nEnter no of elements: ");
    scanf("%d",&n);
    printf("\nEnter array Elements: ");
    for(i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    bubble_sort(a,n);
    printf("\nSorted Array is: ");
    for ( i = 0; i < n; i++)
    {
        printf("%d",a[i]);
        putchar(' ');
    }
}
void bubble_sort(int a[],int n){
    int i,j,temp,flag;
    flag=1;
    for ( i = 1; i < n && flag==1; i++)
    {
        flag=0;
    for(j=0; j<n-i;j++){
        if(a[j]>a[j+1]){
            flag=1;
            temp=a[j];
            a[j]=a[j+1];
            a[j+1]=temp;
}}}}
```

Output for Bubble Sort:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

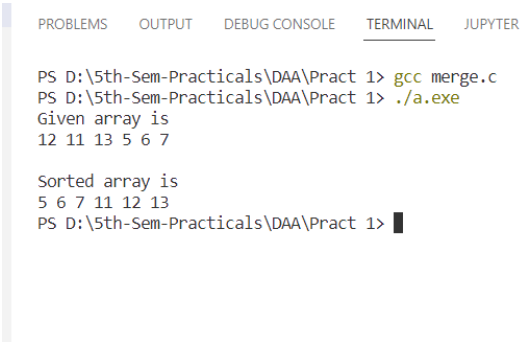
```
PS D:\5th-Sem-Practicals\DAA\Pract 1> gcc bubble.c
PS D:\5th-Sem-Practicals\DAA\Pract 1> ./a.exe

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 > ...

```c
#include <stdio.h>
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}
void printArray(int A[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}
int main()
{
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);
    printf("Given array is \n");
    printArray(arr, arr_size);
    mergeSort(arr, 0, arr_size - 1);
    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}
```

Output for Merge Sort:



Program for Quick Sort:

DAA > Pract 1 > quick.c > swap(int *, int *)

```c
#include <stdio.h>
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}
int partition(int array[], int low, int high) {
  int pivot = array[high];
  int i = (low - 1);
  for (int j = low; j < high; j++) {
    if (array[j] <= pivot) {
      i++;
      swap(&array[i], &array[j]);
    }
  }
  swap(&array[i + 1], &array[high]);
  return (i + 1);
}
void quickSort(int array[], int low, int high) {
  if (low < high) {
  int pi = partition(array, low, high);
 quickSort(array, low, pi - 1);
    quickSort(array, pi + 1, high);
  }
}
void printArray(int array[], int size) {
  for (int i = 0; i < size; ++i) {
    printf("%d  ", array[i]);
  }
  printf("\n");
}
int main() {
  int data[] = {8, 7, 2, 1, 0, 9, 6};
  int n = sizeof(data) / sizeof(data[0]);
  printf("Unsorted Array\n");
  printArray(data, n);
  quickSort(data, 0, n - 1);
  printf("Sorted array: \n");
  printArray(data, n);
}
```

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>
```