1. Write a program to sort the array elements using Merge Sort Technique.

```c
#include <stdio.h>
#include <stdlib.h>



void merge(int arr[], int a, int b, int c){

    int i, j, k;
    int n1 = b - a + 1;
    int n2 = c - b;



    int L[n1], R[n2];



    for (i = 0; i < n1; i++)
        L[i] = arr[a + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[b + 1 + j];



    i = 0;
    j = 0;
    k = a;
    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 a, int c){

    if (a < c) {
        int b = a + (c - a) / 2;


        mergeSort(arr, a, b);
        mergeSort(arr, b + 1, c);

        merge(arr, a, b, c);
    }
}


int main(){

    int arr[] = {38, 27, 43, 10};
    int arr_size = sizeof(arr) / sizeof(arr[0]);

    mergeSort(arr, 0, arr_size - 1);
    int i;
    for (i = 0; i < arr_size; i++)
        printf("%d ", arr[i]);
    printf("\n");

    return 0;
}
```

OUTPUT

```
PS C:\Users\KIIT0001\DAA  LAB> cd "c:\Users\KIIT0001\DAA  LAB\" ; if ($?) { gcc merge_sort.c -o merge_sort } ; if ($?) { .\
merge_sort }
10 27 38 43
PS C:\Users\KIIT0001\DAA  LAB>
```

2. Write a program to sort the array elements using Bucket Sort Technique.

```c
#include <stdio.h>

#define MAX 100
#define SIZE 10

void bucketSort(int arr[], int n) {
    int bucket[MAX] = {0};
    for (int i = 0; i < n; i++)
        bucket[arr[i]]++;


    int idx = 0;
    for (int i = 0; i < MAX; i++) {
        while (bucket[i] > 0) {
            arr[idx++] = i;
            bucket[i]--;
        }
    }
}

int main() {
    int arr[SIZE] = {29, 25, 3, 49, 9, 37, 21, 43, 17, 13};
    int n = SIZE;

    bucketSort(arr, n);

    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");

    return 0;
}
```

OUTPUT

```
PS C:\Users\KIIT0001\DAA  LAB> cd "c:\Users\KIIT0001\DAA  LAB\" ; if ($?) { gcc bucket_sort.c -o bucket_sort } ; if ($?) {
.\bucket_sort }
Sorted array: 3 9 13 17 21 25 29 37 43 49
PS C:\Users\KIIT0001\DAA  LAB>
```

3. Write a program to sort the array elements using Quick Sort Technique.

```c
#include <stdio.h>

void swap(int* a, int* b);

int partition(int arr[], int low, int high) {


    int pivot = arr[high];


    int i = low - 1;


    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }


    swap(&arr[i + 1], &arr[high]);
    return i + 1;
}


void quickSort(int arr[], int low, int high) {
    if (low < high) {


        int pi = partition(arr, low, high);
```

```
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

void swap(int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
}

int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
    int n = sizeof(arr) / sizeof(arr[0]);

    quickSort(arr, 0, n - 1);
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

OUTPUT

```
PS C:\Users\KIIT0001\DAA  LAB> cd "c:\Users\KIIT0001\DAA  LAB\" ; if ($?) { gcc quick_sort.c -o quick_sort } ; if ($?) { .\
quick_sort }
1 5 7 8 9 10
PS C:\Users\KIIT0001\DAA  LAB>
```