Design and Analysis of Algorithms - Lab

Experiment 2

NAME: Ritik Kumar

SAP ID: 590017256

PROGRAM: B.Tech CSE

BATCH: 34

SUBMITTED TO: Mr. Aryan Gupta

GitHub Repository Link:

https://github.com/Ritik2807/DAA_LAB_2_Ritik_Kumar_590017256.git

Objective: Implement merge sort using any of the programming language (C, C++, Java). Make one pdf, including code, 10 different test cases and their screenshots. Also attach plagiarism report in the end.

Source Code (C):

```
#include <stdio.h>

void merge(int arr[], int low, int mid, int high)
{
   int n1 = mid - low + 1;
   int n2 = high - mid;

   int Left[n1], Right[n2];
   for (int i = 0; i < n1; i++)
      Left[i] = arr[low + i];
   for (int i = 0; i < n2; i++)
      Right[i] = arr[mid + 1 + i];
   int left = 0, right = 0, k = low;

   while (left < n1 && right < n2)
   {
      if (Left[left] <= Right[right])
      {
         arr[k] = Left[left];
         left++;
         k++;
      }
      else
      {
         arr[k] = Right[right];
         k++;
         right++;
      }
   }
   while (left < n1)
      arr[k++] = Left[left++];
   while (right < n2)
```

```
        arr[k++] = Right[right++];
}

void mergeSort(int arr[], int low, int high)
{
    if (low < high)
    {
        int mid = low + (high - low) / 2;

        mergeSort(arr, low, mid);
        mergeSort(arr, mid + 1, high);

        merge(arr, low, mid, high);
    }
}
void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main()
{
    int n;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    int low = 0, high = n - 1;

    printf("Original Array: ");
    printArray(arr, n);

    mergeSort(arr, low, high);

    printf("Sorted Array:   ");
    printArray(arr, n);
```

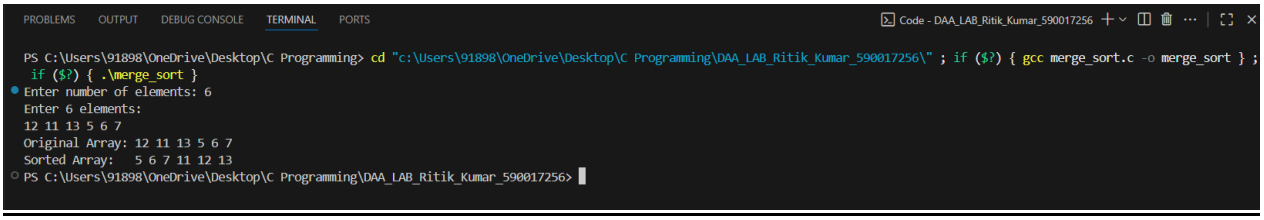
```
    return 0;
}
```

DAA_LAB_Ritik_Kumar_590017256 > merge_sort.c > main()

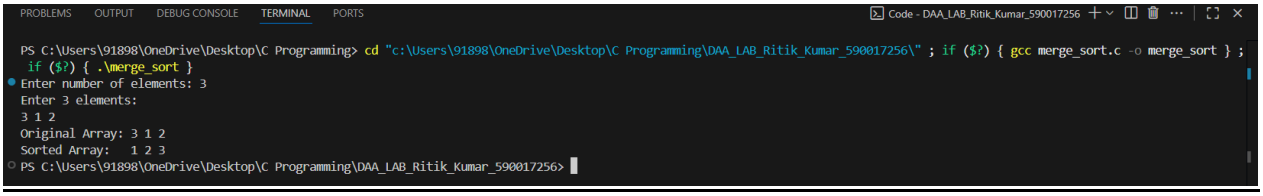
```c
#include <stdio.h>

void merge(int arr[], int low, int mid, int high)
{
    int n1 = mid - low + 1;
    int n2 = high - mid;

    int Left[n1], Right[n2];
    for (int i = 0; i < n1; i++)
        Left[i] = arr[low + i];
    for (int i = 0; i < n2; i++)
        Right[i] = arr[mid + 1 + i];
    int left = 0, right = 0, k = low;

    while (left < n1 && right < n2)
    {
        if (Left[left] <= Right[right])
        {
            arr[k] = Left[left];
            left++;
            k++;
        }
        else
        {
            arr[k] = Right[right];
            k++;
            right++;
        }
    }
    while (left < n1)
        arr[k++] = Left[left++];
    while (right < n2)
        arr[k++] = Right[right++];
}

void mergeSort(int arr[], int low, int high)
{
    if (low < high)
    {
        int mid = low + (high - low) / 2;

        mergeSort(arr, low, mid);
        mergeSort(arr, mid + 1, high);

        merge(arr, low, mid, high);
    }
}
void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main()
{
    int n;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    int low = 0, high = n - 1;

    printf("Original Array: ");
    printArray(arr, n);

    mergeSort(arr, low, high);

    printf("Sorted Array:   ");
    printArray(arr, n);

    return 0;
}
```

All 10 Test cases:

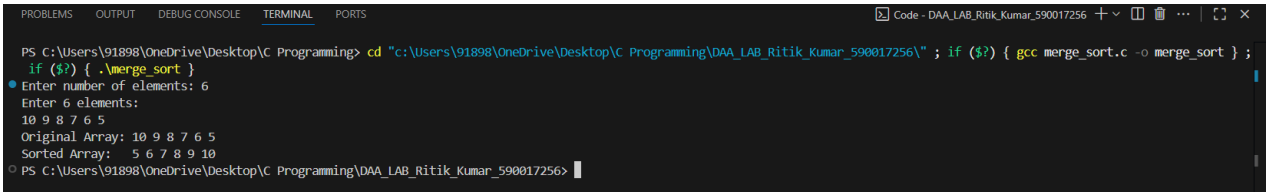
Test Case 1 (Random numbers):



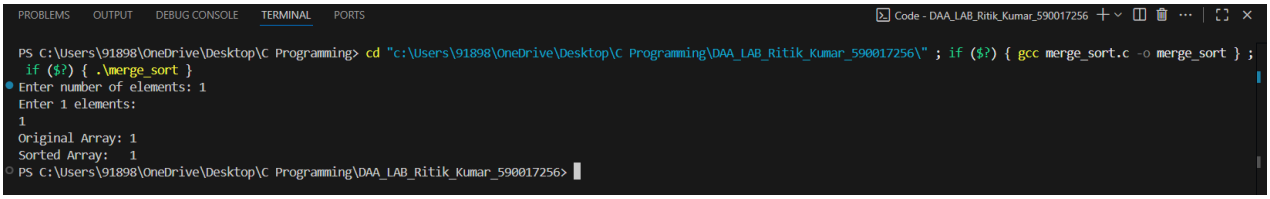
Test Case 2 (Small array, shuffled):



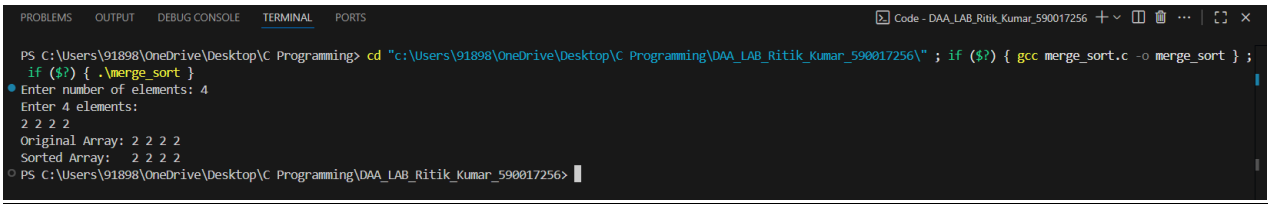
Test Case 3 (Reverse sorted array):



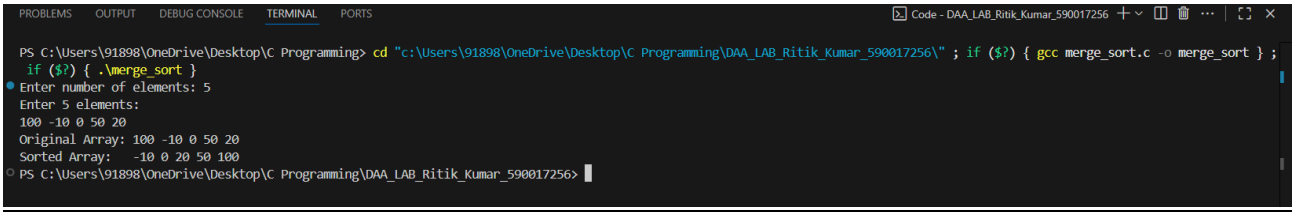
Test Case 4 (Single element):



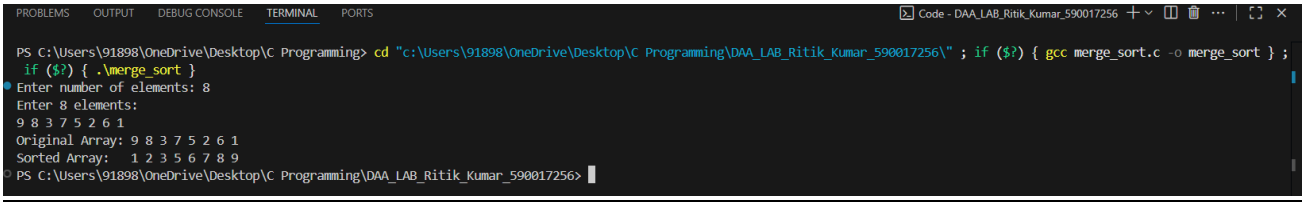
Test Case 5 (All equal elements):



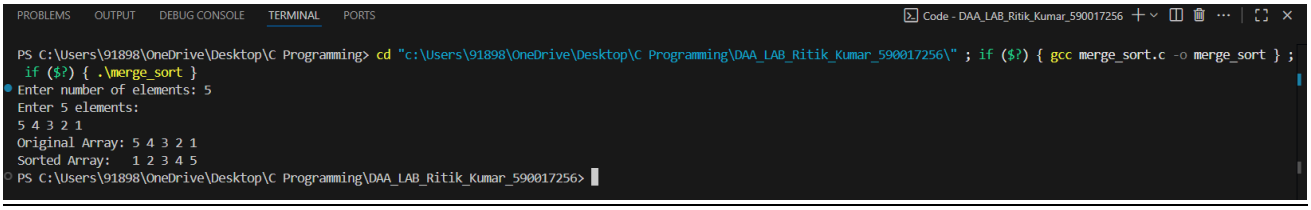
Test Case 6 (Mix of positive and negative):



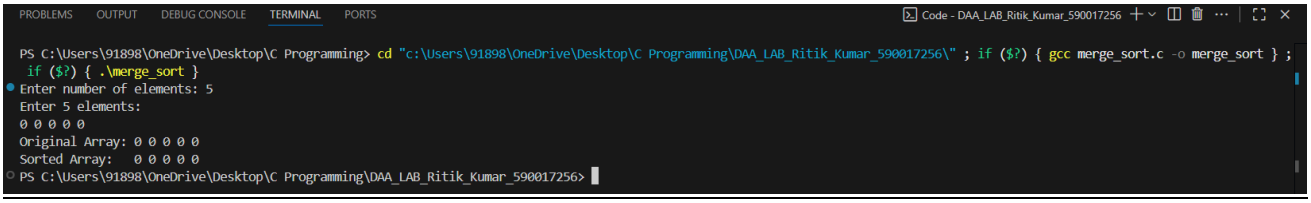
Test Case 7 (More elements, shuffled):



Test Case 8 (Descending order):



Test Case 9 (All zeros):



Test Case 10 (Mixed numbers, random):