

Rajalakshmi Engineering College

Name: Tarrun B

Email: 241801291@rajalakshmi.edu.in

Roll no: 241801291

Phone: 9840381059

Branch: REC

Department: I AI & DS FD

Batch: 2028

Degree: B.E - AI & DS

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 1

Attempt : 1

Total Mark : 10

Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John and Mary are collaborating on a project that involves data analysis. They each have a set of age data, one sorted in ascending order and the other in descending order. However, their analysis requires the data to be in ascending order.

Write a program to help them merge the two sets of age data into a single sorted array in ascending order using merge sort.

Input Format

The first line of input consists of an integer N, representing the number of age values in each dataset.

The second line consists of N space-separated integers, representing the ages of participants in John's dataset (in ascending order).

The third line consists of N space-separated integers, representing the ages of participants in Mary's dataset (in descending order).

Output Format

The output prints a single line containing space-separated integers, which represents the merged dataset of ages sorted in ascending order.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

1 3 5 7 9

10 8 6 4 2

Output: 1 2 3 4 5 6 7 8 9 10

Answer

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

```
// You are using GCC
```

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

```
/* ----- helper: merge two sorted halves ----- */
```

```
void merge(int arr[], int left[], int right[],  
           int left_size, int right_size)
```

```
{
```

```
    int i = 0, j = 0, k = 0;
```

```
    /* copy the smaller element from either left or right into arr */
```

```
    while (i < left_size && j < right_size)
```

```
    {
```

```
        if (left[i] <= right[j])
```

```
            arr[k++] = left[i++];
```

```
        else
```

```
            arr[k++] = right[j++];
```

```
    }
```

```
    /* copy any remaining elements */
```

```
    while (i < left_size)
```

```

arr[k++] = left[i++];
while (j < right_size)
    arr[k++] = right[j++];
}

/* ----- classic merge-sort ----- */
void mergeSort(int arr[], int size)
{
    if (size < 2)      /* 0 or 1 element  already sorted */
        return;

    int mid = size / 2;

    /* create temporary arrays for left and right halves */
    int left[mid];
    int right[size - mid];

    /* copy data into the temporary halves */
    for (int i = 0; i < mid; ++i)
        left[i] = arr[i];
    for (int i = mid; i < size; ++i)
        right[i - mid] = arr[i];

    /* recursively sort the two halves */
    mergeSort(left, mid);
    mergeSort(right, size - mid);

    /* merge the two sorted halves back into arr */
    merge(arr, left, right, mid, size - mid);
}

```

```

int main() {
    int n, m;
    scanf("%d", &n);
    int arr1[n], arr2[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr1[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr2[i]);
    }
}

```

```
}  
int merged[n + n];  
mergeSort(arr1, n);  
mergeSort(arr2, n);  
merge(merged, arr1, arr2, n, n);  
for (int i = 0; i < n + n; i++) {  
    printf("%d ", merged[i]);  
}  
return 0;  
}
```

Status : Correct

Marks : 10/10