Rajalakshmi Engineering College

Name: Kamalesh CT
Email: 240801144@rajalakshmi.edu.in
Roll no: 2116240801144
Phone: 9791302534
Branch: REC
Department: I ECE FB
Batch: 2028
Degree: B.E - ECE

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>


void merge(int merged[], int arr1[], int arr2[], int n1, int n2) {
    int i = 0, j = 0, k = 0;

    // Merge the two sorted arrays into the merged array
    while (i < n1 && j < n2) {
        if (arr1[i] < arr2[j]) {
            merged[k++] = arr1[i++];
        } else {
            merged[k++] = arr2[j++];
        }
    }

    // If any elements remain in arr1, add them to merged
    while (i < n1) {
        merged[k++] = arr1[i++];
    }

    // If any elements remain in arr2, add them to merged
```

```
    while (j < n2) {
        merged[k++] = arr2[j++];
    }
}

void mergeSort(int arr[], int n) {
    if (n <= 1) {
        return;
    }

    int mid = n / 2;
    int left[mid];
    int right[n - mid];

    for (int i = 0; i < mid; i++) {
        left[i] = arr[i];
    }
    for (int i = mid; i < n; i++) {
        right[i - mid] = arr[i];
    }

    mergeSort(left, mid);
    mergeSort(right, n - mid);

    // Merge the sorted left and right subarrays
    int i = 0, j = 0, k = 0;
    while (i < mid && j < n - mid) {
        if (left[i] < right[j]) {
            arr[k++] = left[i++];
        } else {
            arr[k++] = right[j++];
        }
    }

    // If any elements remain in left, add them to arr
    while (i < mid) {
        arr[k++] = left[i++];
    }

    // If any elements remain in right, add them to arr
    while (j < n - mid) {
        arr[k++] = right[j++];
```

```
    }
}
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***