Rajalakshmi Engineering College

Name: Padma Priya D
Email: 240701377@rajalakshmi.edu.in
Roll no: 240701377
Phone: 8668123104
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_CY_Updated

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Aryan is participating in a coding competition where he needs to sort a list of numbers using an efficient sorting algorithm. He decides to use Merge Sort, a divide-and-conquer algorithm, to achieve this. Given a list of n elements, Aryan must implement merge sort to arrange the numbers in ascending order.

Help Aryan by implementing the merge sort algorithm to correctly sort the given list of numbers.

Input Format

The first line of input contains an integer n, the number of elements in the list.

The second line contains n space-separated integers representing the elements

of the list.

Output Format

The output prints the sorted list of numbers in ascending order, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
80 40 20 50 30
Output: 20 30 40 50 80

Answer

```
// You are using GCC
#include <stdio.h>
void merge(int arr[], int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;

    int left[n1], right[n2];
    for (int i = 0; i < n1; i++)
        left[i] = arr[l + i];
    for (int j = 0; j < n2; j++)
        right[j] = arr[m + 1 + j];

    int i = 0, j = 0, k = l;
    while (i < n1 && j < n2) {
        if (left[i] <= right[j])
            arr[k++] = left[i++];
        else
            arr[k++] = right[j++];
    }
    while (i < n1)
        arr[k++] = left[i++];
    while (j < n2)
        arr[k++] = right[j++];
}
```

```
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);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[50];

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

    mergeSort(arr, 0, n - 1);

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

    return 0;
}
```

Status : Correct **Marks : 10/10**

2. Problem Statement

Priya, a data analyst, is working on a dataset of integers. She needs to find the maximum difference between two successive elements in the sorted version of the dataset. The dataset may contain a large number of integers, so Priya decides to use QuickSort to sort the array before finding the difference. Can you help Priya solve this efficiently?

Input Format

The first line of input consists of an integer n, representing the size of the array.

The second line consists of n space-separated integers, representing the elements of the array.

Output Format

The output prints a single integer, representing the maximum difference between two successive elements in the sorted form of the array.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
10

Output: Maximum gap: 0

Answer

```
// You are using GCC
#include <stdio.h>
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
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);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[10];

    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    quickSort(arr, 0, n - 1);
    if (n == 1) {
        printf("Maximum gap: 0\n");
        return 0;
    }
    int max_gap = 0;
    for (int i = 1; i < n; i++) {
        int gap = arr[i] - arr[i - 1];
        if (gap > max_gap) {
            max_gap = gap;
        }
    }

    printf("Maximum gap: %d\n", max_gap);

    return 0;
}
```

Status : Correct **Marks : 10/10**

3. Problem Statement

Reshma is passionate about sorting algorithms and has recently learned

about the merge sort algorithm. She wants to implement a program that utilizes the merge sort algorithm to sort an array of integers, both positive and negative, in ascending order.

Help her in implementing the program.

Input Format

The first line of input consists of an integer N, representing the number of elements in the array.

The second line of input consists of N space-separated integers, representing the elements of the array.

Output Format

The output prints N space-separated integers, representing the array elements sorted in ascending order.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 9
5 -3 0 12 7 -8 2 1 6
Output: -8 -3 0 1 2 5 6 7 12

Answer

```
// You are using GCC
#include <stdio.h>

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

    int L[n1], R[n2];

    for(int i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for(int j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];
```

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

    while(i < n1 && j < n2) {
        if(L[i] <= R[j]) {
            arr[k++] = L[i++];
        } else {
            arr[k++] = R[j++];
        }
    }

    while(i < n1) arr[k++] = L[i++];
    while(j < n2) arr[k++] = R[j++];
}

void mergeSort(int arr[], int left, int right) {
    if(left < right) {
        int mid = left + (right - left)/2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid+1, right);
        merge(arr, left, mid, right);
    }
}

int main() {
    int N;
    scanf("%d", &N);

    int arr[25];
    for(int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    mergeSort(arr, 0, N-1);

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

    return 0;
}
```

Status : Correct *Marks : 10/10*