Rajalakshmi Engineering College

Name: naveen prasath
Email: 240701352@rajalakshmi.edu.in
Roll no: 240701352
Phone: 9585322006
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

Meera is organizing her art supplies, which are represented as a list of integers: red (0), white (1), and blue (2). She needs to sort these supplies so that all items of the same color are adjacent, in the order red, white, and blue. To achieve this efficiently, Meera decides to use QuickSort to sort the items. Can you help Meera arrange her supplies in the desired order?

Input Format

The first line of input consists of an integer n, representing the number of items in the list.

The second line consists of n space-separated integers, where each integer is either 0 (red), 1 (white), or 2 (blue).

Output Format

The output prints the sorted list of integers in a single line, where integers are arranged in the order red (0), white (1), and blue (2).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
2 0 2 1 1 0
Output: Sorted colors:
0 0 1 1 2 2

Answer

```
#include <stdio.h>

void sortColors(int nums[], int n) {
    int low = 0, mid = 0, high = n - 1;
    while (mid <= high) {
        if (nums[mid] == 0) {
            // Swap nums[low] and nums[mid]
            int temp = nums[low];
            nums[low] = nums[mid];
            nums[mid] = temp;
            low++;
            mid++;
        } else if (nums[mid] == 1) {
            mid++;
        } else {
            // Swap nums[mid] and nums[high]
            int temp = nums[mid];
            nums[mid] = nums[high];
            nums[high] = temp;
            high--;
        }
    }
}

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

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

    sortColors(nums, n);

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

    return 0;
}
```

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

2. 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

```
#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];
            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 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[n];
    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**

3. Problem Statement

Sheela wants to distribute cookies to her children, but each child will only be happy if the cookie size meets or exceeds their individual greed factor. She has a limited number of cookies and wants to make as many children happy as possible. Priya decides to sort both the greed factors and cookie sizes using QuickSort to efficiently match cookies with children. Your task is to help Sheela determine the maximum number of children that can be made happy.

Input Format

The first line of input consists of an integer n, representing the number of children.

The second line contains n space-separated integers, where each integer represents the greed factor of a child.

The third line contains an integer m, representing the number of cookies.

The fourth line contains m space-separated integers, where each integer represents the size of a cookie.

Output Format

The output prints a single integer, representing the maximum number of children that can be made happy.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 2 3
2
1 1
Output: The child with greed factor: 1

Answer

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

int compare(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

int findContentChildren(int greed[], int cookies[], int n, int m) {
    qsort(greed, n, sizeof(int), compare);
    qsort(cookies, m, sizeof(int), compare);

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

```
    int count = 0;

    while (i < n && j < m) {
        if (cookies[j] >= greed[i]) {
            count++;
            i++;
        }
        j++;
    }

    return count;
}

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

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

    int result = findContentChildren(greed, cookies, n, m);
    printf("The child with greed factor: %d\n", result);

    return 0;
}
```

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