Rajalakshmi Engineering College

Name: Rayvan Sanjai
Email: 240701425@rajalakshmi.edu.in
Roll no: 2116240701425
Phone: 9380572043
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

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

```c
// 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; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
```

```c
        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 maxHappyChildren(int greed[], int n, int cookies[], int m) {
        quickSort(greed, 0, n - 1);
        quickSort(cookies, 0, m - 1);

        int count = 0, i = 0, j = 0;

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

        return count;
    }

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

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

        scanf("%d", &m);

        int cookies[m];
        for (int i = 0; i < m; i++)
```

```
    scanf("%d", &cookies[i]);

    printf("The child with greed factor: %d\n", maxHappyChildren(greed, n, cookies,
m));

    return 0;
}
```

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

2. 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

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

void sortColors(int arr[], int n) {
    int low = 0, mid = 0, high = n - 1;

    while (mid <= high) {
        if (arr[mid] == 0) {
            int temp = arr[low];
            arr[low++] = arr[mid];
            arr[mid++] = temp;
        } else if (arr[mid] == 1) {
            mid++;
        } else {
            int temp = arr[mid];
            arr[mid] = arr[high];
            arr[high--] = temp;
        }
    }
}

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

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

    sortColors(arr, n);

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

    printf("\n");
    return 0;
}
```

Status : Correct   *Marks : 10/10*

3. 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

```c
// 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];
```

```c
    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[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**