Rajalakshmi Engineering College

Name: Navaneetha Krishnan
Email: 241901067@rajalakshmi.edu.in
Roll no: 241901067
Phone: 8939010233
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

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

Marie, the teacher, wants her students to implement the ascending order of numbers while also exploring the concept of prime numbers.

Students need to write a program that sorts an array of integers using the merge sort algorithm while counting and returning the number of prime integers in the array. Help them to complete the program.

Input Format

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

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

Output Format

The first line of output prints the sorted array of integers in ascending order.

The second line prints the number of prime integers in the array.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
5 3 6 8 9 7 4
Output: Sorted array: 3 4 5 6 7 8 9
Number of prime integers: 3

Answer

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

int isPrime(int n) {
    if (n < 2) return 0;
    if (n % 2 == 0) return n == 2;
    int limit = (int)sqrt(n);
    for (int i = 3; i <= limit; i += 2)
        if (n % i == 0)
            return 0;
    return 1;
}

void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int L[10], R[10];

    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) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

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

    int arr[10];
    for (int i = 0; i < N; i++)
        scanf("%d", &arr[i]);

    int primeCount = 0;
    for (int i = 0; i < N; i++)
        if (isPrime(arr[i]))
            primeCount++;

    mergeSort(arr, 0, N - 1);

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

```
    printf("Number of prime integers: %d\n", primeCount);
    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 quickSort(int arr[], int low, int high) {
```

```
    if (low < high) {
        int pivot = arr[high], i = low - 1;
        for (int j = low; j < high; j++) {
            if (arr[j] <= pivot) {
                i++;
                int tmp = arr[i]; arr[i] = arr[j]; arr[j] = tmp;
            }
        }
        int tmp = arr[i + 1]; arr[i + 1] = arr[high]; arr[high] = tmp;
        int p = i + 1;
        quickSort(arr, low, p - 1);
        quickSort(arr, p + 1, high);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int nums[10];
    for (int i = 0; i < n; i++)
        scanf("%d", &nums[i]);

    quickSort(nums, 0, n - 1);

    int maxGap = 0;
    for (int i = 1; i < n; i++) {
        int gap = nums[i] - nums[i - 1];
        if (gap > maxGap)
            maxGap = gap;
    }

    printf("Maximum gap: %d", maxGap);
    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

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

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pivot = arr[high], i = low - 1;
```

```
        for (int j = low; j < high; j++) {
            if (arr[j] < pivot) {
                i++;
                int tmp = arr[i];
                arr[i] = arr[j];
                arr[j] = tmp;
            }
        }
        int tmp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = tmp;
        int p = i + 1;
        quickSort(arr, low, p - 1);
        quickSort(arr, p + 1, high);
    }
}

int main() {
    int n, m;
    scanf("%d", &n);
    int greed[100];
    for (int i = 0; i < n; i++) {
        scanf("%d", &greed[i]);
    }

    scanf("%d", &m);
    int cookies[100];
    for (int i = 0; i < m; i++) {
        scanf("%d", &cookies[i]);
    }

    quickSort(greed, 0, n - 1);
    quickSort(cookies, 0, m - 1);

    int i = 0, j = 0, happy = 0;
    while (i < n && j < m) {
        if (cookies[j] >= greed[i]) {
            happy++;
            i++;
            j++;
        } else {
            j++;
```

```
        }
    }

    printf("The child with greed factor: %d\n", happy);
    return 0;
}
```

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