Rajalakshmi Engineering College

Name: SUBASH R
Email: 240701538@rajalakshmi.edu.in
Roll no: 240701538
Phone: 9150202710
Branch: REC
Department: I CSE FE
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

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

```
#include <stdio.h>
#include <stdbool.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 i = 0; i < n2; i++) {
        R[i] = arr[mid + 1 + i];
    }

    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++;
    }
}

bool isPrime(int num) {
    if (num <= 1) {
        return false;
    }
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) {
            return false;
        }
    }
    return true;
}

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]);
    }

    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

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>

void swap(int *a, int *b) {
   int t = *a;
   *a = *b;
   *b = t;
}

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 num_children, num_cookies;

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

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

   quickSort(greed_factors, 0, num_children - 1);
   quickSort(cookie_sizes, 0, num_cookies - 1);

   int child_index = 0;
   int cookie_index = 0;
   int happy_children = 0;

   while (child_index < num_children && cookie_index < num_cookies) {
       if (cookie_sizes[cookie_index] >= greed_factors[child_index]) {
           happy_children++;
           child_index++;
       }
       cookie_index++;
   }

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

   return 0;
}
```

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

3. 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

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

void swap(int *a, int *b) {
   int t = *a;
   *a = *b;
   *b = t;
}

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

   quickSort(arr, 0, n - 1);
   printf("Sorted colors:\n");
   for (int i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   return 0;
}
```

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