Rajalakshmi Engineering College

Name: SHIVANISREE K B
Email: 240701501@rajalakshmi.edu.in
Roll no: 240701501
Phone: 7358464804
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

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

// Function to check if a number is prime
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;
}

// Merge function for merge sort
void merge(int arr[], int left, int mid, int right) {
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;

    int L[10], R[10];  // As per constraints, max size is 10

    for (i = 0; i < n1; i++)
```

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

    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++];
}

// Merge sort function
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[10];  // As per constraints, maximum N is 10
    for (int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    // Count prime numbers
```

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

    // Sort the array using merge sort
    mergeSort(arr, 0, N - 1);

    // Output the sorted array
    printf("Sorted array:");
    for (int i = 0; i < N; i++) {
        printf(" %d", arr[i]);
    }
    printf("\n");

    // Output the number of primes
    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>

// Function to swap two elements
void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Partition function for QuickSort
int partition(int arr[], int low, int high) {
    int pivot = arr[high];  // Choosing the last element as pivot
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }

    swap(&arr[i + 1], &arr[high]);
    return i + 1;
}

// QuickSort function
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);
    }
}

// Function to find the maximum gap
int maximumGap(int arr[], int n) {
    if (n < 2)
        return 0;

    quickSort(arr, 0, n - 1);

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

    return maxGap;
}

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

    int arr[10];  // Constraint: n ≤ 10
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int result = maximumGap(arr, n);
    printf("Maximum gap: %d\n", result);

    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

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

// Merge function
void merge(int arr[], int left, int mid, int right) {
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;
```

```
    int L[50], R[50];  // Temporary arrays

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

    i = 0; j = 0; k = left;

    // Merging the temporary arrays back into arr[]
    while (i < n1 && j < n2) {
        if (L[i] <= R[j])
            arr[k++] = L[i++];
        else
            arr[k++] = R[j++];
    }

    // Copy remaining elements of L[], if any
    while (i < n1)
        arr[k++] = L[i++];

    // Copy remaining elements of R[], if any
    while (j < n2)
        arr[k++] = R[j++];
}

// Merge Sort function
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        // Sort first and second halves
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        // Merge the sorted halves
        merge(arr, left, mid, right);
    }
}

int main() {
```

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

    int arr[50];  // According to constraints, max size is 50

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

    // Apply merge sort
    mergeSort(arr, 0, n - 1);

    // Print the sorted array
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");

    return 0;
}
```

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