Rajalakshmi Engineering College

Name: Pooja S
Email: 240701386@rajalakshmi.edu.in
Roll no: 240701386
Phone: 8838480229
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

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];
    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);
}

// 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 maximum gap between successive elements
int findMaxGap(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[100];
    for(int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

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

    return 0;
}
```

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

2. 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 n1 = mid - left + 1;
    int n2 = right - mid;

    int L[10], R[10]; // Size limited to 10 as per constraints

    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)
        arr[k++] = (L[i] <= R[j]) ? L[i++] : 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) / 2;

        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

        merge(arr, left, mid, right);
    }
}
```

```
// Main function
int main() {
    int n, arr[10], primeCount = 0;

    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[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("\nNumber of prime integers: %d\n", primeCount);

    return 0;
}
```

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

3. 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

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

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

    int L[25], R[25];  // max size 25 as per constraints

    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++];
}

// Recursive merge sort function
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[25]; // max size 25 as per constraints
    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**