Rajalakshmi Engineering College

Name: Prithika S
Email: 240701400@rajalakshmi.edu.in
Roll no: 240701400
Phone: 9790212894
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

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>

// Function to check if a number is prime
int isPrime(int n) {
    if (n < 2) return 0;
    int limit = (int)sqrt(n);
    for (int i = 2; i <= limit; i++) {
        if (n % i == 0)
            return 0;
    }
    return 1;
}

// Merge function to merge two sorted halves
void merge(int arr[], int left[], int right[], int left_size, int right_size) {
    int i = 0, j = 0, k = 0;
    while (i < left_size && j < right_size) {
        if (left[i] < right[j])
            arr[k++] = left[i++];
        else
            arr[k++] = right[j++];
    }
```

```
    while (i < left_size)
        arr[k++] = left[i++];
    while (j < right_size)
        arr[k++] = right[j++];
}

// Merge sort function
void mergeSort(int arr[], int size) {
    if (size < 2)
        return;

    int mid = size / 2;
    int left[mid], right[size - mid];

    for (int i = 0; i < mid; i++)
        left[i] = arr[i];
    for (int i = mid; i < size; i++)
        right[i - mid] = arr[i];

    mergeSort(left, mid);
    mergeSort(right, size - mid);
    merge(arr, left, right, mid, size - mid);
}

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

    // Sort the array using merge sort
    mergeSort(arr, n);

    // Count prime numbers
    int primeCount = 0;
    for (int i = 0; i < n; i++) {
        if (isPrime(arr[i]))
            primeCount++;
    }

    // Print the sorted array
```

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

    // Print the count of prime integers
    printf("Number of prime integers: %d\n", primeCount);

    return 0;
}
```

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

2. 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>

// Function to merge two sorted halves into arr[]
void merge(int arr[], int left[], int right[], int left_size, int right_size) {
    int i = 0, j = 0, k = 0;

    while(i < left_size && j < right_size) {
        if(left[i] <= right[j])
            arr[k++] = left[i++];
        else
            arr[k++] = right[j++];
    }

    while(i < left_size)
        arr[k++] = left[i++];

    while(j < right_size)
        arr[k++] = right[j++];
}

// Recursive merge sort function
void mergeSort(int arr[], int size) {
    if(size < 2)  // Base case: 1 or 0 elements is already sorted
        return;

    int mid = size / 2;
    int left[mid], right[size - mid];

    // Copy data into left and right subarrays
    for(int i = 0; i < mid; i++)
        left[i] = arr[i];
    for(int i = mid; i < size; i++)
        right[i - mid] = arr[i];

    // Recursive calls to sort the subarrays
    mergeSort(left, mid);
```

```
    mergeSort(right, size - mid);

    // Merge sorted halves back into arr
    merge(arr, left, right, mid, size - mid);
}

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

    int arr[50];
    for(int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    mergeSort(arr, n);

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

    return 0;
}
```

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

3. Problem Statement

Ravi is given an array of integers and is tasked with sorting it in a unique way. He needs to sort the elements in such a way that the elements at odd positions are in descending order, and the elements at even positions are in ascending order. Ravi decided to use the Insertion Sort algorithm for this task.

Your task is to help ravi, to create even_odd_insertion_sort function to sort the array as per the specified conditions and then print the sorted array.

Example

Input:

10

25 36 96 58 74 14 35 15 75 95

Output:

96 14 75 15 74 36 35 58 25 95

Input Format

The first line of input consists of a single integer, N, which represents the size of the array.

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

Output Format

The output displays the sorted array using the even-odd insertion sort algorithm and prints the sorted array.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
3 1 4 2
Output: 4 1 3 2

Answer

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

void even_odd_insertion_sort(int arr[], int n) {
    // Sort odd positions (1-based) in descending order
    for (int i = 2; i < n; i += 2) {  // i = index of odd positions (0-based is even index)
        int key = arr[i];
        int j = i - 2;
        while (j >= 0 && arr[j] < key) {
```

```
            arr[j + 2] = arr[j];
            j -= 2;
        }
        arr[j + 2] = key;
    }

    // Sort even positions (1-based) in ascending order
    for (int i = 3; i < n; i += 2) {  // i = index of even positions (0-based is odd index)
        int key = arr[i];
        int j = i - 2;
        while (j >= 1 && arr[j] > key) {
            arr[j + 2] = arr[j];
            j -= 2;
        }
        arr[j + 2] = key;
    }
}

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

    even_odd_insertion_sort(arr, n);

    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

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