Rajalakshmi Engineering College

Name: KEERTHI PRIYA T
Email: 240701258@rajalakshmi.edu.in
Roll no: 2116240701258
Phone: 7397397221
Branch: REC
Department: I CSE FC
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 : 20

Section 1 : Coding

1. 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) {
    int i, j, key;

    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        // Check parity
        if (i % 2 == 0) {
            // Even index: sort ascending among even indices only
```

```
            while (j >= 0 && j % 2 == 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j--;
            }
        } else {
            // Odd index: sort descending among odd indices only
            while (j >= 0 && j % 2 == 1 && arr[j] < key) {
                arr[j + 1] = arr[j];
                j--;
            }
        }
        arr[j + 1] = key;
    }
}

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

    int arr[n];
    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]);

    return 0;
}
```

Status : Wrong **Marks : 0/10**

2. 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>

// 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];  // pivot
    int i = low - 1;  // Index of smaller element

    for (int j = low; j <= high - 1; j++) {
        // If current element is smaller or equal to pivot
        if (arr[j] <= pivot) {
```

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

// QuickSort recursive function
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        // pi is partitioning index, arr[pi] is now at right place
        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: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");

    return 0;
}
```

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

3. 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>

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

```
    return true;
}

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 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 - 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]);

    mergeSort(arr, 0, n-1);

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

    printf("Sorted array:");
    for (int i=0; i<n; i++) {
        printf(" %d", arr[i]);
    }
    printf("  Number of prime integers: %d\n", primeCount);

    return 0;
}
```

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