Rajalakshmi Engineering College

Name: Pranav Narayanan
Email: 240701393@rajalakshmi.edu.in
Roll no:
Phone: 9500579427
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

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 insertion_sort(int arr[], int n, int ascending) {
    int i, j, key;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        if (ascending) {
            // Sort ascending
            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j--;
```

```
        }
    } else {
        // Sort descending
        while (j >= 0 && arr[j] < key) {
            arr[j + 1] = arr[j];
            j--;
        }
    }
    arr[j + 1] = key;
  }
}

void even_odd_insertion_sort(int arr[], int n) {
  int odd_pos_elements[10], even_pos_elements[10];
  int odd_count = 0, even_count = 0;
  int i;

  // Separate odd and even positions (1-indexed)
  // odd positions => indices 0, 2, 4,... (0-based)
  for (i = 0; i < n; i += 2) {
    odd_pos_elements[odd_count++] = arr[i];
  }
  // even positions => indices 1, 3, 5,...
  for (i = 1; i < n; i += 2) {
    even_pos_elements[even_count++] = arr[i];
  }

  // Sort odd positions descending
  insertion_sort(odd_pos_elements, odd_count, 0);

  // Sort even positions ascending
  insertion_sort(even_pos_elements, even_count, 1);

  // Merge back
  for (i = 0; i < odd_count; i++) {
    arr[2 * i] = odd_pos_elements[i];
  }
  for (i = 0; i < even_count; i++) {
    arr[2 * i + 1] = even_pos_elements[i];
  }

  // Print the sorted array
```

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

int main() {
    int n, i;
    int arr[10];

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

    even_odd_insertion_sort(arr, n);

    return 0;
}
```

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

2. 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>

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 as per constraints (N <= 25)
    int i, j, k;

    // Copy data to temporary arrays L[] and R[]
    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; // Initial index of first subarray
    j = 0; // Initial index of second subarray
    k = left; // Initial index of merged subarray

    // Merge the temp arrays back into arr[left..right]
    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++];
    }
}

void merge_sort(int arr[], int left, int right) {
    if (left < right) {
        int mid = (left + right) / 2;

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

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

int main() {
    int n, i;
    int arr[25];  // max size as per constraints

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

    merge_sort(arr, 0, n - 1);

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

    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>

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

```
    int j, temp;

    for (j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            // Swap arr[i] and arr[j]
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    // Swap arr[i+1] and arr[high] (pivot)
    temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    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);
    }
}

int main() {
    int n, i;
    int arr[100];  // max size constraint

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

    quickSort(arr, 0, n - 1);

    printf("Sorted colors: ");
    for (i = 0; i < n; i++) {
```

```
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

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