Rajalakshmi Engineering College

Name: Sneha Raju R
Email: 240701519@rajalakshmi.edu.in
Roll no: 240701519
Phone: 7550004064
Branch: REC
Department: 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

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>

// Function to merge two halves
void merge(int arr[], int left, int mid, int right) {
   int i, j, k;
   int n1 = mid - left + 1;
   int n2 = right - mid;

   // Temporary arrays
   int L[n1], R[n2];

   // Copy data to temp 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];

   // Merge the temp arrays back into arr[]
   i = 0; j = 0; k = left;

   while (i < n1 && j < n2) {
      if (L[i] <= R[j]) {
         arr[k] = L[i];
         i++;
      } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

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

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

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

// Main function
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);

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

    return 0;
}
```

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

2. 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>

// Insertion Sort Ascending
void insertionSortAsc(int arr[], int n) {
   for (int i = 1; i < n; i++) {
      int key = arr[i];
      int j = i - 1;
      while (j >= 0 && arr[j] > key) {
         arr[j + 1] = arr[j];
         j--;
      }
      arr[j + 1] = key;
   }
}

// Insertion Sort Descending
void insertionSortDesc(int arr[], int n) {
   for (int i = 1; i < n; i++) {
      int key = arr[i];
      int j = i - 1;
      while (j >= 0 && arr[j] < key) {
         arr[j + 1] = arr[j];
```

```
            j--;
        }
        arr[j + 1] = key;
    }
}

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

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

    // Separate elements by position type
    int oddPos[n], evenPos[n];
    int oddCount = 0, evenCount = 0;

    for (int i = 0; i < n; i++) {
        if ((i + 1) % 2 == 1) { // 1-based odd position
            oddPos[oddCount++] = arr[i];
        } else {
            evenPos[evenCount++] = arr[i];
        }
    }

    // Sort odd-position elements in descending order
    insertionSortDesc(oddPos, oddCount);

    // Sort even-position elements in ascending order
    insertionSortAsc(evenPos, evenCount);

    // Reconstruct the array
    int oddIndex = 0, evenIndex = 0;
    for (int i = 0; i < n; i++) {
        if ((i + 1) % 2 == 1) {
            arr[i] = oddPos[oddIndex++];
        } else {
            arr[i] = evenPos[evenIndex++];
        }
    }

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

    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>

void sortColors(int arr[], int n) {
   int low = 0, mid = 0, high = n - 1;

   while (mid <= high) {
      if (arr[mid] == 0) {
         // Swap arr[low] and arr[mid]
         int temp = arr[low];
         arr[low] = arr[mid];
         arr[mid] = temp;
         low++;
         mid++;
      }
      else if (arr[mid] == 1) {
         mid++;
      }
      else { // arr[mid] == 2
         // Swap arr[mid] and arr[high]
         int temp = arr[mid];
         arr[mid] = arr[high];
         arr[high] = temp;
         high--;
      }
   }
}

int main() {
   int n;
   scanf("%d", &n);
   int arr[100];

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

   sortColors(arr, n);
```

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

    return 0;
}
```

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