Rajalakshmi Engineering College

Name: Phaveen S
Email: 240701383@rajalakshmi.edu.in
Roll no: 240701383
Phone: null
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 uniquely. 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.

Your task is to help Ravi create a program that uses insertion sort to sort the array as per the specified conditions and then print the sorted array. Position starts from 1.

Example

Input:

Size of the array = 10

Array elements = 25 36 96 58 74 14 35 15 75 95

Output:

Resultant array = 96 14 75 15 74 36 35 58 25 95

Explanation:

Initial Array: 25 36 96 58 74 14 35 15 75 95

Elements at odd positions (1, 3, 5, 7, 9): 25 96 74 35 75

Elements at odd positions sorted descending order: 96 75 74 35 25

Elements at even positions (2, 4, 6, 8, 10): 36 58 14 15 95

Elements at even positions sorted ascending order: 14 15 36 58 95

So, the final array is 96 14 75 15 74 36 35 58 25 95.

Input Format

The first line contains an integer N, representing the number of elements in the array.

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

Output Format

The output displays integers, representing the sorted array elements separated by a space.

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

        while (j >= 0 && ((ascending && arr[j] > key) || (!ascending && arr[j] < key))) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[10];

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

    int odd[10], even[10];
    int oddCount = 0, evenCount = 0;

    for (int i = 0; i < n; i++) {
        if ((i + 1) % 2 == 1) {
            odd[oddCount++] = arr[i];
        } else {
            even[evenCount++] = arr[i];
        }
    }

    insertionSort(odd, oddCount, 0);
    insertionSort(even, evenCount, 1);

    int oddIndex = 0, evenIndex = 0;
    for (int i = 0; i < n; i++) {
        if ((i + 1) % 2 == 1) {
            printf("%d ", odd[oddIndex++]);
        } else {
            printf("%d ", even[evenIndex++]);
        }
    }

    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>

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

   int L[50], R[50];

   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;
   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, arr[50];

    scanf("%d", &n);

    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]);
    }

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

        while (j >= 0 && ((ascending && arr[j] > key) || (!ascending && arr[j] < key))) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

void even_odd_insertion_sort(int arr[], int n) {
    int odd[10], even[10];
    int oddCount = 0, evenCount = 0;

    for (int i = 0; i < n; i++) {
        if ((i + 1) % 2 == 1) {
            odd[oddCount++] = arr[i];
        } else {
            even[evenCount++] = arr[i];
        }
    }

    insertionSort(odd, oddCount, 0);
    insertionSort(even, evenCount, 1);

    int oddIndex = 0, evenIndex = 0;
    for (int i = 0; i < n; i++) {
        if ((i + 1) % 2 == 1) {
            printf("%d ", odd[oddIndex++]);
        } else {
            printf("%d ", even[evenIndex++]);
        }
    }
    printf("\n");
}

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

    scanf("%d", &n);

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

    return 0;
}
```

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