Rajalakshmi Engineering College

Name: MRIDHULA DEVI M
Email: 240701337@rajalakshmi.edu.in
Roll no: 240701337
Phone: 9840329629
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

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[n1], R[n2];

   // Copy data to temporary arrays L[] and R[]
   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;
   // Merge the temporary arrays back into arr[left..right]
   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++;
    }
}

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 the sorted halves
        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);

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

    return 0;
}
```

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

2. 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 reverse) {
   int i, key, j;
   for (i = 1; i < n; i++) {
      key = arr[i];
      j = i - 1;
      if (reverse) {
         // Sort in descending order
         while (j >= 0 && arr[j] < key) {
            arr[j + 1] = arr[j];
            j = j - 1;
         }
      } else {
         // Sort in ascending order
         while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
            }
        }
        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]);
    }

    // Separate elements at odd and even positions
    int oddCount = (N + 1) / 2;
    int evenCount = N / 2;
    int odd[oddCount], even[evenCount];

    for (int i = 0; i < N; i++) {
        if (i % 2 == 0) {
            odd[i / 2] = arr[i];
        } else {
            even[i / 2] = arr[i];
        }
    }

    // Sort odd-positioned elements in descending order
    insertionSort(odd, oddCount, 1);

    // Sort even-positioned elements in ascending order
    insertionSort(even, evenCount, 0);

    // Reconstruct the array
    int oddIndex = 0, evenIndex = 0;
    for (int i = 0; i < N; i++) {
        if (i % 2 == 0) {
            arr[i] = odd[oddIndex++];
        } else {
            arr[i] = even[evenIndex++];
        }
    }
    // 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 insertionSort(int arr[], int n, int reverse) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        if (reverse) {
            // Sort in descending order
            while (j >= 0 && arr[j] < key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
        } else {
            // Sort in ascending order
            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
        }
        arr[j + 1] = key;
    }
}
```

```
void even_odd_insertion_sort(int arr[], int N) {
   int oddCount = (N + 1) / 2;
   int evenCount = N / 2;
   int odd[oddCount], even[evenCount];

   // Separate elements into odd and even indexed arrays
   for (int i = 0; i < N; i++) {
      if (i % 2 == 0) {
         odd[i / 2] = arr[i];
      } else {
         even[i / 2] = arr[i];
      }
   }

   // Sort odd-indexed elements in descending order
   insertionSort(odd, oddCount, 1);

   // Sort even-indexed elements in ascending order
   insertionSort(even, evenCount, 0);

   // Reconstruct the original array
   int oddIndex = 0, evenIndex = 0;
   for (int i = 0; i < N; i++) {
      if (i % 2 == 0) {
         arr[i] = odd[oddIndex++];
      } else {
         arr[i] = even[evenIndex++];
      }
   }
}

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]);
   }
   printf("\n");

   return 0;
}
```

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