Rajalakshmi Engineering College

Name: eswar rs
Email: 240801074@rajalakshmi.edu.in
Roll no: 240801074
Phone: 7845648127
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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

```
#include <stdio.h>
```

```
// Function to perform insertion sort
void insertionSort(int arr[], int n, int descending) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && (descending ? arr[j] < key : arr[j] > key)) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

// Function to sort the array uniquely as per given conditions
void customSort(int arr[], int n) {
    int oddCount = (n + 1) / 2;  // Number of elements at odd positions
    int evenCount = n / 2;       // Number of elements at even positions
    int odd[oddCount], even[evenCount];

    // Extract elements based on positions
    int oddIdx = 0, evenIdx = 0;
    for (int i = 0; i < n; i++) {
        if (i % 2 == 0) {
            odd[oddIdx++] = arr[i];
        } else {
            even[evenIdx++] = arr[i];
        }
    }

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

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

    // Merge sorted values back into the original array structure
    oddIdx = 0, evenIdx = 0;
    for (int i = 0; i < n; i++) {
        if (i % 2 == 0) {
            arr[i] = odd[oddIdx++];
        } else {
```

```
            arr[i] = even[evenIdx++];
        }
    }
}

// Main function
int main() {
    int N;
    scanf("%d", &N);
    int arr[N];

    // Reading array elements
    for (int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    // Sort the array
    customSort(arr, N);

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

    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;

    // 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[left..right]
    i = 0; // Initial index of first subarray
    j = 0; // Initial index of second subarray
```

```
    k = left; // Initial index of merged subarray

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

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

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

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[50];  // Maximum constraint is n = 50
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    mergeSort(arr, 0, n - 1);
```

```
    // Output 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

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 swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high]; // pivot
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if (arr[j] <= pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }

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

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;
    scanf("%d", &n);

    int arr[100];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
```

```
    quicksort(arr, 0, n - 1);

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

    return 0;
}
```

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