Rajalakshmi Engineering College

Name: Sowmyalakshmi N
Email: 240701524@rajalakshmi.edu.in
Roll no: 240701524
Phone: 9003767185
Branch: REC
Department: I 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

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

```c
#include <stdio.h>

// Function to perform insertion sort on specific indices
void insertion_sort(int arr[], int indices[], int count, int ascending) {
    for (int i = 1; i < count; i++) {
        int key = arr[indices[i]];
        int j = i - 1;

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

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

    int arr[10]; // Since 1 ≤ N ≤ 10
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Arrays to store indices of odd and even positions (1-based)
    int odd_indices[10], even_indices[10];
    int odd_count = 0, even_count = 0;

    // Separate indices based on position
    for (int i = 0; i < n; i++) {
        if ((i + 1) % 2 == 1) { // 1-based odd position
            odd_indices[odd_count++] = i;
        } else { // 1-based even position
            even_indices[even_count++] = i;
        }
    }

    // Sort odd-position elements in descending order
    insertion_sort(arr, odd_indices, odd_count, 0);
```

```
    // Sort even-position elements in ascending order
    insertion_sort(arr, even_indices, even_count, 1);

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

    return 0;
}
```

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

2. 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>

// Function to swap two elements
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Partition function for QuickSort
int partition(int arr[], int low, int high) {
    int pivot = arr[high]; // choosing the last element as 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);
}

// QuickSort function
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pivotIndex = partition(arr, low, high);

        quickSort(arr, low, pivotIndex - 1);
        quickSort(arr, pivotIndex + 1, high);
    }
}

int main() {
    int n;
```

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

    int arr[100]; // 1 ≤ n ≤ 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]);
    }

    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>

// Function to perform insertion sort on odd/even indexed elements
void even_odd_insertion_sort(int arr[], int n) {
    // Sort elements at odd positions (0-based: 0, 2, 4...) in descending order
    for (int i = 2; i < n; i += 2) {
        int key = arr[i];
        int j = i - 2;
        while (j >= 0 && arr[j] < key) {
            arr[j + 2] = arr[j];
            j -= 2;
        }
        arr[j + 2] = key;
    }

    // Sort elements at even positions (0-based: 1, 3, 5...) in ascending order
    for (int i = 3; i < n; i += 2) {
        int key = arr[i];
```

```
        int j = i - 2;
        while (j >= 1 && arr[j] > key) {
            arr[j + 2] = arr[j];
            j -= 2;
        }
        arr[j + 2] = key;
    }

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

int main() {
    int n;
    scanf("%d", &n);
    int arr[10]; // since 1 ≤ N ≤ 10
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    even_odd_insertion_sort(arr, n);
    return 0;
}
```

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