Rajalakshmi Engineering College

Name: NISHANTH B
Email: 240701364@rajalakshmi.edu.in
Roll no: 240701364
Phone: 7904264876
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

Priya, a data analyst, is working on a dataset of integers. She needs to find the maximum difference between two successive elements in the sorted version of the dataset. The dataset may contain a large number of integers, so Priya decides to use QuickSort to sort the array before finding the difference. Can you help Priya solve this efficiently?

Input Format

The first line of input consists of an integer n, representing the size of the array.

The second line consists of n space-separated integers, representing the elements of the array.

Output Format

The output prints a single integer, representing the maximum difference between two successive elements in the sorted form of the array.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
10

Output: Maximum gap: 0

Answer

```
// You are using GCC
#include <stdio.h>

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    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[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    quickSort(arr, 0, n - 1);
    int maxGap = 0;
    for (int i = 1; i < n; i++) {
        int gap = arr[i] - arr[i - 1];
        if (gap > maxGap) {
            maxGap = gap;
        }
    }
    printf("Maximum gap: %d\n", maxGap);
    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 insertionSortAsc(int arr[], int n) {
   for (int i = 1; i < n; i++) {
        int key = arr[i], j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

void insertionSortDesc(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i], 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], odd[n], even[n], oddCount = 0, evenCount = 0;
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
        if ((i + 1) % 2 == 1)
            odd[oddCount++] = arr[i];
        else
            even[evenCount++] = arr[i];
    }
    insertionSortDesc(odd, oddCount);
    insertionSortAsc(even, evenCount);
    int o = 0, e = 0;
    for (int i = 0; i < n; i++) {
        if ((i + 1) % 2 == 1)
            printf("%d ", odd[o++]);
        else
            printf("%d ", even[e++]);
    }
    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>

int partition(int arr[], int low, int high) {
   int pivot = arr[high];
   int i = low - 1;
   for (int j = low; j < high; j++) {
       if (arr[j] < pivot) {
           i++;
           int temp = arr[i];
           arr[i] = arr[j];
           arr[j] = temp;
       }
   }
   int temp = arr[i + 1];
   arr[i + 1] = arr[high];
   arr[high] = temp;
   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[n];
   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***