Rajalakshmi Engineering College

Name: Sajine Santhakumar
Email: 240701459@rajalakshmi.edu.in
Roll no: 240701459
Phone: 9952076750
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

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.

240701459

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>

// Function to merge two subarrays
void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;  // size of left subarray
    int n2 = right - mid;     // size of right subarray

    int L[n1], R[n2];

    // Copy data to temp arrays
    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 temp arrays back into arr[]
    while (i < n1 && j < n2) {
        if (L[i] <= R[j])
            arr[k++] = L[i++];
        else
            arr[k++] = R[j++];
    }
```

240701459

```
    // Copy remaining elements of L[]
    while (i < n1)
        arr[k++] = L[i++];

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

// Recursive function to perform merge sort
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, arr[25];

    // Input array size
    scanf("%d", &n);

    // Input array elements
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    // Perform merge sort
    mergeSort(arr, 0, n - 1);

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

    return 0;
```
240701459

}

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

2. Problem Statement

Sheela wants to distribute cookies to her children, but each child will only be happy if the cookie size meets or exceeds their individual greed factor. She has a limited number of cookies and wants to make as many children happy as possible. Priya decides to sort both the greed factors and cookie sizes using QuickSort to efficiently match cookies with children. Your task is to help Sheela determine the maximum number of children that can be made happy.

Input Format

The first line of input consists of an integer n, representing the number of children.

The second line contains n space-separated integers, where each integer represents the greed factor of a child.

The third line contains an integer m, representing the number of cookies.

The fourth line contains m space-separated integers, where each integer represents the size of a cookie.

Output Format

The output prints a single integer, representing the maximum number of children that can be made happy.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 2 3
2
1 1

240701459

Output: The child with greed factor: 1

Answer

```
#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];
   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, m;
   scanf("%d", &n);
   int greed_factors[n];
   for(int i = 0; i < n; i++) {
      scanf("%d", &greed_factors[i]);
   }
   scanf("%d", &m);
   int cookie_sizes[m];
   for(int i = 0; i < m; i++) {
      scanf("%d", &cookie_sizes[i]);
   }
```

240701459

```
    quickSort(greed_factors, 0, n - 1);
    quickSort(cookie_sizes, 0, m - 1);
    int child_index = 0, cookie_index = 0;
    int happy_children = 0;
    while(child_index < n && cookie_index < m) {
        if(cookie_sizes[cookie_index] >= greed_factors[child_index]) {
            happy_children++;
            child_index++;
            cookie_index++;
        } else {
            cookie_index++;
        }
    }
    printf("The child with greed factor: %d\n", happy_children);
    return 0;
}
```

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

3. 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

240701459

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>

// Insertion sort for ascending order
void insertionSortAsc(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
```

240701459

```
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

// Insertion sort for descending order
void insertionSortDesc(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int 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[10], odd[10], even[10];
    int 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);
```
240701459

```
    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***

240701459