Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_MCQ_Updated_1

Attempt : 1
Total Mark : 20
Marks Obtained : 13

Section 1 : MCQ

1. Is Merge Sort a stable sorting algorithm?

Answer

Yes, always stable.

Status : Correct ***Marks : 1/1***

2. Which of the following is true about Quicksort?

Answer

It is an in-place sorting algorithm

Status : Correct ***Marks : 1/1***

3. Which of the following strategies is used to improve the efficiency of Quicksort in practical implementations?

Answer

Sorting the array in reverse order before applying Quicksort

Status : Wrong ***Marks : 0/1***

4. What happens during the merge step in Merge Sort?

Answer

Two sorted subarrays are combined into one sorted array

Status : Correct ***Marks : 1/1***

5. What happens when Merge Sort is applied to a single-element array?

Answer

The array is divided and merged as usual

Status : Wrong ***Marks : 0/1***

6. Which of the following modifications can help Quicksort perform better on small subarrays?

Answer

Switching to Insertion Sort for small subarrays

Status : Correct ***Marks : 1/1***

7. In a quick sort algorithm, what role does the pivot element play?

Answer

It is used to partition the array

Status : Correct ***Marks : 1/1***

8. Let P be a quick sort program to sort numbers in ascending order using the first element as a pivot. Let t1 and t2 be the number of comparisons made by P for the inputs {1, 2, 3, 4, 5} and {4, 1, 5, 3, 2}, respectively. Which one of the following holds?

Answer

t1 = t2

Status : Wrong ***Marks : 0/1***

9. Which of the following scenarios is Merge Sort preferred over Quick Sort?

Answer

When sorting linked lists

Status : Correct ***Marks : 1/1***

10. The following code snippet is an example of a quick sort. What do the 'low' and 'high' parameters represent in this code?

```
void quickSort(int arr[], int low, int high) {
   if (low < high) {
      int pivot = partition(arr, low, high);
      quickSort(arr, low, pivot - 1);
      quickSort(arr, pivot + 1, high);
   }
}
```

Answer

The range of elements to sort within the array

Status : Correct ***Marks : 1/1***

11. Which of the following statements is true about the merge sort algorithm?

Answer

It requires additional memory for merging

Status : Correct **Marks : 1/1**

12. Why is Merge Sort preferred for sorting large datasets compared to Quick Sort?

Answer

Merge Sort has better worst-case time complexity

Status : Correct **Marks : 1/1**

13. Which of the following is not true about QuickSort?

Answer

An in-place algorithm

Status : Wrong **Marks : 0/1**

14. Consider the Quick Sort algorithm, which sorts elements in ascending order using the first element as a pivot. Then which of the following input sequences will require the maximum number of comparisons when this algorithm is applied to it?

Answer

52 25 76 67 89

Status : Wrong **Marks : 0/1**

15. What is the main advantage of Quicksort over Merge Sort?

Answer

Quicksort is stable

Status : Wrong **Marks : 0/1**

16. Which of the following methods is used for sorting in merge sort?

Answer

merging

Status : Correct ***Marks : 1/1***

17. Merge sort is _____.

Answer

Comparison-based sorting algorithm

Status : Correct ***Marks : 1/1***

18. In a quick sort algorithm, where are smaller elements placed to the pivot during the partition process, assuming we are sorting in increasing order?

Answer

To the left of the pivot

Status : Correct ***Marks : 1/1***

19. What is the best sorting algorithm to use for the elements in an array that are more than 1 million in general?

Answer

Quick sort.

Status : Correct ***Marks : 1/1***

20. Which of the following sorting algorithms is based on the divide and conquer method?

Answer

Selection Sort

Status : Wrong ***Marks : 0/1***

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John and Mary are collaborating on a project that involves data analysis. They each have a set of age data, one sorted in ascending order and the other in descending order. However, their analysis requires the data to be in ascending order.

Write a program to help them merge the two sets of age data into a single sorted array in ascending order using merge sort.

Input Format

The first line of input consists of an integer N, representing the number of age values in each dataset.

The second line consists of N space-separated integers, representing the ages of participants in John's dataset (in ascending order).

The third line consists of N space-separated integers, representing the ages of participants in Mary's dataset (in descending order).

Output Format

The output prints a single line containing space-separated integers, which represents the merged dataset of ages sorted in ascending order.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 3 5 7 9
10 8 6 4 2

Output: 1 2 3 4 5 6 7 8 9 10

Answer

```
#include <stdio.h>

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

// Function to merge two sorted arrays
void mergeSortedArrays(int arr1[], int arr2[], int merged[], int N) {
    int i = 0, j = N - 1, k = 0;

    // Merge in ascending order by reversing arr2's traversal
    while (i < N && j >= 0) {
        if (arr1[i] < arr2[j]) {
            merged[k++] = arr1[i++];
        } else {
            merged[k++] = arr2[j--];
        }
    }

    // Copy remaining elements from arr1 (if any)
    while (i < N) {
        merged[k++] = arr1[i++];
    }
```

```
    // Copy remaining elements from arr2 (if any)
    while (j >= 0) {
        merged[k++] = arr2[j--];
    }
}

int main() {
    int N;
    scanf("%d", &N); // Read the number of elements

    int johnAges[N], maryAges[N], mergedAges[2 * N];

    // Read John's dataset (ascending order)
    for (int i = 0; i < N; i++) {
        scanf("%d", &johnAges[i]);
    }

    // Read Mary's dataset (descending order)
    for (int i = 0; i < N; i++) {
        scanf("%d", &maryAges[i]);
    }

    // Merge the datasets
    mergeSortedArrays(johnAges, maryAges, mergedAges, N);

    // Print the merged sorted array
    for (int i = 0; i < 2 * N; i++) {
        printf("%d ", mergedAges[i]);
    }

    return 0;
}
int main() {
    int n, m;
    scanf("%d", &n);
    int arr1[n], arr2[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr1[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr2[i]);
```

```
    }
    int merged[n + n];
    mergeSort(arr1, n);
    mergeSort(arr2, n);
    merge(merged, arr1, arr2, n, n);
    for (int i = 0; i < n + n; i++) {
        printf("%d ", merged[i]);
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Nandhini asked her students to arrange a set of numbers in ascending order. She asked the students to arrange the elements using insertion sort, which involves taking each element and placing it in its appropriate position within the sorted portion of the array.

Assist them in the task.

Input Format

The first line of input consists of the value of n, representing the number of array elements.

The second line consists of n elements, separated by a space.

Output Format

The output prints the sorted array, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
67 28 92 37 59

Output: 28 37 59 67 92

Answer

```
#include <stdio.h>

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

// Function to perform insertion sort
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        // Shift elements of arr[0..i-1] that are greater than key to one position ahead
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

// Main function to handle input and output
int main() {
    int n;

    // Reading number of elements
    scanf("%d", &n);

    int arr[n];  // Declare array of size n
```

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

    // Sorting the array using insertion sort
    insertionSort(arr, n);

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

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

    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are the lead developer of a text-processing application that assists writers in organizing their thoughts. One crucial feature is a character-sorting service that helps users highlight the most critical elements of their text.

To achieve this, you decide to enhance the service to sort characters in descending order using the Quick-Sort algorithm. Implement the algorithm to efficiently rearrange the characters, ensuring that it is sorted in descending order.

Input Format

The first line of the input consists of a positive integer value N, representing the number of characters to be sorted.

The second line of input consists of N space-separated lowercase alphabetical characters.

Output Format

The output displays the set of alphabetical characters, sorted in descending order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
a d g j k
Output: k j g d a

Answer

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

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

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

// Partition function for Quick Sort
int partition(char arr[], int low, int high) {
    char pivot = arr[high];  // Choosing last element as pivot
    int i = low - 1; // Index for the smaller element

    for (int j = low; j < high; j++) {
        if (arr[j] > pivot) { // Sorting in descending order
            i++;
            swap(&arr[i], &arr[j]);
        }
```

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

// Quick Sort function
void quickSort(char 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;

    // Read the number of characters
    scanf("%d", &N);

    char arr[N]; // Declare character array

    // Read space-separated lowercase characters
    for (int i = 0; i < N; i++) {
        scanf(" %c", &arr[i]); // Space before %c to ignore leading whitespaces
    }

    // Sort the characters using Quick Sort
    quickSort(arr, 0, N - 1);

    // Print sorted characters in descending order
    for (int i = 0; i < N; i++) {
        printf("%c ", arr[i]);
    }

    return 0;
}
```

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

```
    char characters[n];

    for (int i = 0; i < n; i++) {
        char input;
        scanf(" %c", &input);
        characters[i] = input;
    }

    quicksort(characters, 0, n - 1);

    for (int i = 0; i < n; i++) {
        printf("%c ", characters[i]);
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Kavya, a software developer, is analyzing data trends. She has a list of integers and wants to identify the nth largest number in the list after sorting the array using QuickSort.

To optimize performance, Kavya is required to use QuickSort to sort the list before finding the nth largest number.

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 nums.

The third line consists of an integer k, representing the position of the largest

number you need to print after sorting the array.

Output Format

The output prints the k-th largest number in the sorted array (sorted in ascending order).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
-1 0 1 2 -1 -4
3
Output: 0

Answer

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


// Comparison function for qsort
int compare(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}

// Function to find the k-th largest number
void findNthLargest(int* nums, int n, int k) {
    // Sort the array in ascending order
    qsort(nums, n, sizeof(int), compare);

    // Print the k-th largest element
    printf("%d\n", nums[n - k]);
}

int main() {
    int n, k;
    scanf("%d", &n);
    int* nums = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
```

```
    }
    scanf("%d", &k);
    findNthLargest(nums, n, k);
    free(nums);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Jose has an array of N fractional values, represented as double-point numbers. He needs to sort these fractions in increasing order and seeks your help.

Write a program to help Jose sort the array using the merge sort algorithm.

Input Format

The first line of input consists of an integer N, representing the number of fractions to be sorted.

The second line consists of N double-point numbers, separated by spaces, representing the fractions array.

Output Format

The output prints N double-point numbers, sorted in increasing order, and rounded to three decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
0.123 0.543 0.321 0.789

Output: 0.123 0.321 0.543 0.789

Answer

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


// Function to merge two halves of the array
void merge(double arr[], int left, int mid, int right) {
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;

    double leftArr[n1], rightArr[n2];

    // Copy data to temporary arrays
    for (i = 0; i < n1; i++)
        leftArr[i] = arr[left + i];
    for (j = 0; j < n2; j++)
        rightArr[j] = arr[mid + 1 + j];

    // Merge the temp arrays back into arr[left...right]
    i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (leftArr[i] <= rightArr[j]) {
            arr[k++] = leftArr[i++];
        } else {
            arr[k++] = rightArr[j++];
        }
    }
```

```
    // Copy remaining elements of leftArr, if any
    while (i < n1) {
        arr[k++] = leftArr[i++];
    }

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

// Merge sort function
void mergeSort(double arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        // Recursively 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);
    double fractions[n];
    for (int i = 0; i < n; i++) {
        scanf("%lf", &fractions[i]);
    }
    mergeSort(fractions, 0, n - 1);
    for (int i = 0; i < n; i++) {
        printf("%.3f ", fractions[i]);
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_PAH_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

You are working as a programmer at a sports academy, and the academy holds various sports competitions regularly.

As part of the academy's system, you need to sort the scores of the participants in descending order using the Quick Sort algorithm.

Write a program that takes the scores of n participants as input and uses the Quick Sort algorithm to sort the scores in descending order. Your program should display the sorted scores after the sorting process.

Input Format

The first line of input consists of an integer n, which represents the number of scores.

The second line of input consists of n integers, which represent scores separated by spaces.

Output Format

Each line of output represents an iteration of the Quick Sort algorithm, displaying the elements of the array at that iteration.

After the iterations are complete, the last line of output prints the sorted scores in descending order separated by space.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 5
78 54 96 32 53

Output: Iteration 1: 78 54 96 53 32
Iteration 2: 96 54 78
Iteration 3: 78 54
Sorted Order: 96 78 54 53 32

Answer

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

// Function to partition the array and display iterations
int partition(int arr[], int low, int high, int iteration) {
    int pivot = arr[high]; // Selecting the pivot
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if (arr[j] >= pivot) { // Sorting in descending order
            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;

    // Display the iteration
    printf("Iteration %d: ", iteration);
    for (int k = low; k <= high; k++) {
        printf("%d ", arr[k]);
    }
    printf("\n");

    return i + 1;
}

// Quick Sort function
void quickSort(int arr[], int low, int high, int *iteration) {
    if (low < high) {
        int pi = partition(arr, low, high, (*iteration)++);
        quickSort(arr, low, pi - 1, iteration);
        quickSort(arr, pi + 1, high, iteration);
    }
}

int main() {
    int n;
    scanf("%d", &n); // Read number of scores
    int scores[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &scores[i]); // Read scores
    }

    int iteration = 1;
    quickSort(scores, 0, n - 1, &iteration);

    // Display sorted scores
    printf("Sorted Order: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", scores[i]);
    }
    printf("\n");
```

```
    return 0;
}
```

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

2. Problem Statement

Vishnu, a math enthusiast, is given a task to explore the magic of numbers. He has an array of positive integers, and his goal is to find the integer with the highest digit sum in the sorted array using the merge sort algorithm.

You have to assist Vishnu in implementing the merge sort algorithm.

Input Format

The first line of input consists of an integer N, representing the number of elements in the array.

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

Output Format

The first line of output prints "The sorted array is: " followed by the sorted array, separated by a space.

The second line prints "The integer with the highest digit sum is: " followed by an integer representing the highest-digit sum.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
123 456 789 321 654
Output: The sorted array is: 123 321 456 654 789
The integer with the highest digit sum is: 789

Answer

```
// You are using GCC
```

```c
#include <stdio.h>

// Function to compute digit sum of a number
int digitSum(int num) {
    int sum = 0;
    while (num > 0) {
        sum += num % 10;
        num /= 10;
    }
    return sum;
}

// Merge function for Merge Sort
void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int L[n1], R[n2];

    for (int i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (int i = 0; i < n2; i++)
        R[i] = arr[mid + 1 + i];

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k++] = L[i++];
        } else {
            arr[k++] = R[j++];
        }
    }

    while (i < n1)
        arr[k++] = L[i++];

    while (j < n2)
        arr[k++] = R[j++];
}

// Merge Sort function
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
```

```
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        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]);

    // Sorting the array using Merge Sort
    mergeSort(arr, 0, N - 1);

    // Display sorted array
    printf("The sorted array is: ");
    for (int i = 0; i < N; i++)
        printf("%d ", arr[i]);
    printf("\n");

    // Find the integer with the highest digit sum
    int maxSum = -1, maxNum = arr[0];
    for (int i = 0; i < N; i++) {
        int sum = digitSum(arr[i]);
        if (sum > maxSum) {
            maxSum = sum;
            maxNum = arr[i];
        }
    }

    // Display the integer with the highest digit sum
    printf("The integer with the highest digit sum is: %d\n", maxNum);

    return 0;
}
```

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

3. Problem Statement

You're a coach managing a list of finishing times for athletes in a race. The times are stored in an array, and you need to sort this array in ascending order to determine the rankings.

You'll use the insertion sort algorithm to accomplish this.

Input Format

The first line of input contains an integer n, representing the number of athletes.

The second line contains n space-separated integers, each representing the finishing time of an athlete in seconds.

Output Format

The output prints the sorted finishing times of the athletes in ascending order.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
75 89 65 90 70
Output: 65 70 75 89 90

Answer

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

// Function to perform insertion sort
void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        // Move elements that are greater than key one position ahead
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
```

```
        }
        arr[j + 1] = key;
    }
}

int main() {
    int n;

    // Read number of athletes
    scanf("%d", &n);
    int times[n];

    // Read finishing times
    for (int i = 0; i < n; i++) {
        scanf("%d", &times[i]);
    }

    // Sorting the finishing times using insertion sort
    insertionSort(times, n);

    // Display sorted finishing times
    for (int i = 0; i < n; i++) {
        printf("%d ", times[i]);
    }

    printf("\n");
    return 0;
}
```

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

4. Problem Statement

Alex is working on a project that involves merging and sorting two arrays. He wants to write a program that merges two arrays, sorts the merged array in ascending order, removes duplicates, and prints the sorted array without duplicates.

Help Alex to implement the program using the merge sort algorithm.

Input Format

The first line of input consists of an integer N, representing the number of elements in the first array.

The second line consists of N integers, separated by spaces, representing the elements of the first array.

The third line consists of an integer M, representing the number of elements in the second array.

The fourth line consists of M integers, separated by spaces, representing the elements of the second array.

Output Format

The output prints space-separated integers, representing the merged and sorted array in ascending order, with duplicate elements removed.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
1 2 3 4
3
3 4 5
Output: 1 2 3 4 5

Answer

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

// Merge function for Merge Sort
void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int L[n1], R[n2];

    for (int i = 0; i < n1; i++)
        L[i] = arr[left + i];
```

```
    for (int i = 0; i < n2; i++)
        R[i] = arr[mid + 1 + i];

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k++] = L[i++];
        } else {
            arr[k++] = R[j++];
        }
    }

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

// Merge Sort function
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

// Function to remove duplicates
int removeDuplicates(int arr[], int n) {
    if (n == 0 || n == 1)
        return n;

    int temp[n], j = 0;
    for (int i = 0; i < n - 1; i++) {
        if (arr[i] != arr[i + 1])
            temp[j++] = arr[i];
    }
    temp[j++] = arr[n - 1];

    for (int i = 0; i < j; i++)
        arr[i] = temp[i];
```

```
    return j;
}

int main() {
    int N, M;

    // Read first array
    scanf("%d", &N);
    int arr1[N];
    for (int i = 0; i < N; i++)
        scanf("%d", &arr1[i]);

    // Read second array
    scanf("%d", &M);
    int arr2[M];
    for (int i = 0; i < M; i++)
        scanf("%d", &arr2[i]);

    // Merge both arrays
    int merged[N + M];
    for (int i = 0; i < N; i++)
        merged[i] = arr1[i];
    for (int i = 0; i < M; i++)
        merged[N + i] = arr2[i];

    // Sort the merged array using Merge Sort
    mergeSort(merged, 0, N + M - 1);

    // Remove duplicates
    int newSize = removeDuplicates(merged, N + M);

    // Print the sorted array without duplicates
    for (int i = 0; i < newSize; i++)
        printf("%d ", merged[i]);
    printf("\n");

    return 0;
}
```

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

5. Problem Statement

You are working on an optimization task for a sorting algorithm that uses insertion sort. Your goal is to determine the efficiency of the algorithm by counting the number of swaps needed to sort an array of integers.

Write a program that takes an array as input and calculates the number of swaps performed during the insertion sort process.

Example 1:

Input:

5

2 1 3 1 2

Output:

4

Explanation:

Step 1: [2, 1, 3, 1, 2] (No swaps)

Step 2: [1, 2, 3, 1, 2] (1 swap, element 1 shifts 1 place to the left)

Step 3: [1, 2, 3, 1, 2] (No swaps)

Step 4: [1, 1, 2, 3, 2] (2 swaps; element 1 shifts 2 places to the left)

Step 5: [1, 1, 2, 2, 3] (1 swap, element 2 shifts 1 place to the left)

Total number of swaps: 1 + 2 + 1 = 4

Example 2:

Input:

7

12 15 1 5 6 14 11

Output:

10

Explanation:

Step 1: [12, 15, 1, 5, 6, 14, 11] (No swaps)

Step 2: [12, 15, 1, 5, 6, 14, 11] (1 swap, element 15 shifts 1 place to the left)

Step 3: [12, 15, 1, 5, 6, 14, 11] (No swaps)

Step 4: [1, 12, 15, 5, 6, 14, 11] (2 swaps, element 1 shifts 2 places to the left)

Step 5: [1, 5, 12, 15, 6, 14, 11] (1 swap, element 5 shifts 1 place to the left)

Step 6: [1, 5, 6, 12, 15, 14, 11] (2 swaps, element 6 shifts 2 places to the left)

Step 7: [1, 5, 6, 12, 14, 15, 11] (1 swap, element 14 shifts 1 place to the left)

Step 8: [1, 5, 6, 11, 12, 14, 15] (3 swaps, element 11 shifts 3 places to the left)

Total number of swaps: 1 + 2 + 1 + 2 + 1 + 3 = 10

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 the number of swaps performed during the insertion sort process.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
2 1 3 1 2
Output: 4

Answer

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

// Function to perform insertion sort and count swaps
int insertionSort(int arr[], int n) {
    int swapCount = 0;

    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        // Move elements that are greater than key one position ahead
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
            swapCount++; // Counting swaps
        }
        arr[j + 1] = key;
    }

    return swapCount;
}

int main() {
    int n;

    // Read number of elements
    scanf("%d", &n);
    int arr[n];

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

    // Perform insertion sort and count swaps
    int totalSwaps = insertionSort(arr, n);

    // Print total swaps
    printf("%d\n", totalSwaps);

    return 0;
```

```
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Alagappan ramesh
Email: 241901003@rajalakshmi.edu.in
Roll no: 241901003
Phone: 7845207607
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_CY_Updated

Attempt : 1
Total Mark : 30
Marks Obtained : 20

Section 1 : Coding

1. 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 partition the array
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) { // Sorting in ascending order
            i++;
            // Swap arr[i] and arr[j]
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    // Swap arr[i+1] and pivot
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    return i + 1;
}

// Quick Sort function
```

```
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;

    // Read number of items
    scanf("%d", &n);
    int arr[n];

    // Read item colors
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Sort the array using Quick Sort
    quickSort(arr, 0, n - 1);

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

    return 0;
}
```

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

2. 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>

// Function to partition the array
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) { // Sorting in ascending order
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    // Swap arr[i+1] and pivot
```

```
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    return i + 1;
}

// QuickSort function
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);
    }
}

// Function to find the maximum gap between successive elements
int findMaxGap(int arr[], int n) {
    int maxGap = 0;
    for (int i = 1; i < n; i++) {
        int gap = arr[i] - arr[i - 1];
        if (gap > maxGap) {
            maxGap = gap;
        }
    }
    return maxGap;
}

int main() {
    int n;

    // Read the number of elements
    scanf("%d", &n);
    int arr[n];

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

    // Sort the array using QuickSort
    quickSort(arr, 0, n - 1);
```

```
    // Find the maximum gap between successive elements
    int maxGap = (n > 1) ? findMaxGap(arr, n) : 0;

    // Print the result
    printf("Maximum gap: %d\n", maxGap);

    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 in ascending order
void insertionSortAscending(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;
    }
}

// Function to perform insertion sort in descending order
void insertionSortDescending(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;
    }
}

// Function to sort elements at odd indices in descending order and even indices
in ascending order
void even_odd_insertion_sort(int arr[], int n) {
    int odd[n / 2 + 1], even[n / 2 + 1]; // Separate odd and even indexed elements
    int oddCount = 0, evenCount = 0;

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

    // Sort odd indexed elements in descending order
    insertionSortDescending(odd, oddCount);

    // Sort even indexed elements in ascending order
    insertionSortAscending(even, evenCount);

    // Merge sorted elements back into original array
    int oddIndex = 0, evenIndex = 0;
    for (int i = 0; i < n; i++) {
        if (i % 2 == 0)
            arr[i] = even[evenIndex++];
        else
            arr[i] = odd[oddIndex++];
    }
}

int main() {
    int N;

    // Read the number of elements
    scanf("%d", &N);
    int arr[N];
```

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

    // Perform the sorting based on the specified conditions
    even_odd_insertion_sort(arr, N);

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

    return 0;
}
```

Status : Wrong **Marks : 0/10**