Rajalakshmi Engineering College

Name: Naren Kartic B
Email: 241901065@rajalakshmi.edu.in
Roll no:
Phone: 6374678252
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>

void merge(int arr[], int left[], int right[], int left_size, int right_size) {
   int i = 0, j = 0, k = 0;

   while (i < left_size && j < right_size) {
      if (left[i] <= right[j]) {
         arr[k] = left[i];
         i++;
      } else {
         arr[k] = right[j];
         j++;
      }
      k++;
   }

   while (i < left_size) {
      arr[k] = left[i];
      i++;
      k++;
   }
```

```
    while (j < right_size) {
        arr[k] = right[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int size) {
    if (size > 1) {
        int mid = size / 2;
        int left[mid];
        int right[size - mid];

        for (int i = 0; i < mid; i++) {
            left[i] = arr[i];
        }

        for (int i = mid; i < size; i++) {
            right[i - mid] = arr[i];
        }

        mergeSort(left, mid);
        mergeSort(right, size - mid);
        merge(arr, left, right, mid, size - mid);
    }
}


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: Naren Kartic B
Email: 241901065@rajalakshmi.edu.in
Roll no:
Phone: 6374678252
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>

void insertionSort(int arr[], int n) {
    int i, j, key;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
}

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: Naren Kartic B
Email: 241901065@rajalakshmi.edu.in
Roll no:
Phone: 6374678252
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>


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

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

   for (int j = low; j <= high - 1; j++) {
      if (arr[j] > pivot) {
         i++;
         swap(&arr[i], &arr[j]);
      }
   }
   swap(&arr[i + 1], &arr[high]);
   return (i + 1);
}
```

```
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;
    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: Naren Kartic B
Email: 241901065@rajalakshmi.edu.in
Roll no:
Phone: 6374678252
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>


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 pivot = partition(arr, low, high); // Get the partition index
        quickSort(arr, low, pivot - 1);        // Sort the elements before the pivot
        quickSort(arr, pivot + 1, high);       // Sort the elements after the pivot
    }
}

void findNthLargest(int* nums, int n, int k) {
    quickSort(nums, 0, n - 1); // Use QuickSort to sort the array

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

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: Naren Kartic B
Email: 241901065@rajalakshmi.edu.in
Roll no:
Phone: 6374678252
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>


int compare(double a, double b) {
    return (a < b) ? -1 : (a > b);
}
void merge(double arr[], int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;

    double L[n1], R[n2];
    int i, j, k;

    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    i = 0;
    j = 0;
    k = l;

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

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

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
void mergeSort(double arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;

        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}


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: Naren Kartic B
Email: 241901065@rajalakshmi.edu.in
Roll no:
Phone: 6374678252
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

```
#include <stdio.h>

void sortColors(int nums[], int n) {
    int low = 0, mid = 0, high = n - 1;

    while (mid <= high) {
        if (nums[mid] == 0) {
            // Swap nums[low] and nums[mid]
            int temp = nums[low];
            nums[low] = nums[mid];
            nums[mid] = temp;
            low++;
            mid++;
        } else if (nums[mid] == 1) {
            mid++;
        } else {
            // Swap nums[mid] and nums[high]
            int temp = nums[mid];
            nums[mid] = nums[high];
            nums[high] = temp;
            high--;
        }
    }
}

int main() {
    int n;
    scanf("%d", &n); // Read the number of items in the list
    int nums[n];

    // Read the space-separated integers into the array
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }

    // Sort the colors
    sortColors(nums, n);

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

    return 0;
}
```

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

2. 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.

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

```
#include <stdio.h>

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

   // Create temporary arrays
   int L[n1], R[n2];

   // Copy data to temp arrays L[] and R[]
   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];

   // Merge the temporary arrays back into the original array
   int i = 0, j = 0, k = left;
   while (i < n1 && j < n2) {
      if (L[i] <= R[j]) {
         arr[k] = L[i];
         i++;
      } else {
         arr[k] = R[j];
         j++;
      }
```

```
        k++;
    }

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

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

// Function to implement merge sort
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2; // Find the middle point

        // Recursively sort the 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);  // Read the number of elements in the array
    int arr[N];

    // Read the array elements
    for (int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }
    // Perform merge sort on the array
    mergeSort(arr, 0, N - 1);

    // Print 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

Marie, the teacher, wants her students to implement the ascending order of numbers while also exploring the concept of prime numbers.

Students need to write a program that sorts an array of integers using the merge sort algorithm while counting and returning the number of prime integers in the array. Help them to complete the program.

Input Format

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

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

Output Format

The first line of output prints the sorted array of integers in ascending order.

The second line prints the number of prime integers in the array.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
5 3 6 8 9 7 4

Output: Sorted array: 3 4 5 6 7 8 9
Number of prime integers: 3

Answer

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Naren Kartic B
Email: 241901065@rajalakshmi.edu.in
Roll no:
Phone: 6374678252
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 : 20

Section 1 : MCQ

1. 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

22 25 56 67 89

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

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

Answer

Choosing the pivot randomly or using the median-of-three method

Status : Correct *Marks : 1/1*

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

Answer

It can be implemented as a stable sort

Status : Correct *Marks : 1/1*

4. 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 &gt; t2

Status : Correct *Marks : 1/1*

5. Merge sort is ________.

Answer

Comparison-based sorting algorithm

Status : Correct *Marks : 1/1*

6. 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*

7. 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***

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

Answer

It requires additional memory for merging

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

9. Which of the following is true about Quicksort?

Answer

It is an in-place sorting algorithm

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

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

Answer

When sorting linked lists

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

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

Answer

Merge Sort

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

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

Answer

merging

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

13. 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***

14. 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***

15. Is Merge Sort a stable sorting algorithm?

Answer

Yes, always stable.

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

16. 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***

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

Answer

Quicksort requires less auxiliary space

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

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

Answer

The array remains unchanged and no merging is required

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

19. 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***

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

Answer

Two sorted subarrays are combined into one sorted array

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

Rajalakshmi Engineering College

Name: Naren Kartic B
Email: 241901065@rajalakshmi.edu.in
Roll no:
Phone: 6374678252
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 : 40

Section 1 : Coding

1. 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

```
#include <stdio.h>

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

   int L[20], R[20];  // assuming maximum 10 + 10 elements

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

   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 from sorted array
int removeDuplicates(int arr[], int n, int result[]) {
    if (n == 0)
        return 0;

    int j = 0;
    result[j++] = arr[0];

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

    return j;
}

int main() {
    int N, M, i;
    int arr1[10], arr2[10], merged[20], result[20];

    scanf("%d", &N);
    for (i = 0; i < N; i++)
        scanf("%d", &arr1[i]);

    scanf("%d", &M);
    for (i = 0; i < M; i++)
        scanf("%d", &arr2[i]);

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

    int total = N + M;

    // Sort the merged array
    mergeSort(merged, 0, total - 1);

    // Remove duplicates
    int uniqueCount = removeDuplicates(merged, total, result);

    // Print final output
    for (i = 0; i < uniqueCount; i++) {
        printf("%d", result[i]);
        if (i != uniqueCount - 1)
            printf(" ");
    }

    return 0;
}
```

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

2. 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

```
#include <stdio.h>

int main() {
   int n, i, j, key;
   int times[20];  // max size as per constraints

   // Input number of athletes
   scanf("%d", &n);

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

   // Insertion sort
   for (i = 1; i < n; i++) {
      key = times[i];
      j = i - 1;
```

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

    // Print sorted times
    for (i = 0; i < n; i++) {
        printf("%d", times[i]);
        if (i != n - 1) {
            printf(" ");
        }
    }

    return 0;
}
```

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

3. 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

```
#include <stdio.h>

int main() {
   int n, i, j, key, swapCount = 0;
   int arr[10];  // maximum size as per constraints

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

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

   // Insertion sort with swap counting
   for (i = 1; i < n; i++) {
      key = arr[i];
      j = i - 1;
```

```
        // Count how many shifts (equivalent to swaps)
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];  // shift right
            j--;
            swapCount++;          // count each shift as a swap
        }
        arr[j + 1] = key;         // place key at correct position
    }

    // Print total swap count
    printf("%d", swapCount);

    return 0;
}
```

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

4. 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

```
#include <stdio.h>

// Function to calculate 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 i, j, k;
   int n1 = mid - left + 1;
   int n2 = right - mid;
   int L[10], R[10];  // Maximum size according to constraints

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

   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, i;
    int arr[10];  // Max N = 10

    // Read input
    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    // Sort the array using merge sort
    mergeSort(arr, 0, N - 1);

    // Print sorted array
    printf("The sorted array is: ");
    for (i = 0; i < N; i++) {
        printf("%d", arr[i]);
        if (i != N - 1)
            printf(" ");
    }

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

    // Print highest digit sum number
    printf("\nThe integer with the highest digit sum is: %d", maxNum);

    return 0;
}
```

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

5. 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

```
#include <stdio.h>

// Function to print the array in the specified format
void printArray(int arr[], int low, int high, int isFinal) {
    if (isFinal) {
        printf("Sorted Order: ");
        for (int i = 0; i <= high; i++) {
            printf("%d", arr[i]);
            if (i != high) printf(" ");
        }
    } else {
        printf("Iteration %d: ", ++iterationCount);
        for (int i = low; i <= high; i++) {
            printf("%d", arr[i]);
            if (i != high) printf(" ");
        }
        printf("\n");
    }
}
```

```
// Global iteration counter
int iterationCount = 0;

// Partition function for descending quicksort
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1, temp;

    for (int j = low; j < high; j++) {
        if (arr[j] > pivot) {  // descending order
            i++;
            // Swap arr[i] and arr[j]
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

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

    return i + 1;
}

// QuickSort in descending order with iteration printing
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        printArray(arr, low, high, 0);
        int pi = partition(arr, low, high);

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

int main() {
    int n, arr[10];

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

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

    // Start quicksort
    quickSort(arr, 0, n - 1);

    // Print final sorted array
    printArray(arr, 0, n - 1, 1);

    return 0;
}
```

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