Rajalakshmi Engineering College

Name: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
Branch: REC
Department: l 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. Merge sort is _____.

Answer

Comparison-based sorting algorithm

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

2. 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***

3. 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***

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

Answer

Merge Sort

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

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

Answer

It requires additional memory for merging

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

Answer

When sorting linked lists

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

8. 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***

9. 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***

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

Answer

Two sorted subarrays are combined into one sorted array

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

11. 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***

12. Is Merge Sort a stable sorting algorithm?

Answer

Yes, always stable.

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

13. 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***

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

Answer

merging

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

15. Which of the following is true about Quicksort?

Answer

It is an in-place sorting algorithm

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

16. 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***

17. 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***

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

Answer

Quicksort requires less auxiliary space

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

19. 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 : Correct ***Marks : 1/1***

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

Answer

It can be implemented as a stable sort

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

Rajalakshmi Engineering College

Name: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
Branch: REC
Department: l 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: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
Branch: REC
Department: l 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: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll
no: 241901036
Phone: 7010799637
Branch: REC
Department: l 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: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
Branch: REC
Department: l 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);
      quickSort(arr, low, pivot - 1);
```

```
        quickSort(arr, pivot + 1, high);
    }
}

void findNthLargest(int* nums, int n, int k) {
    quickSort(nums, 0, n - 1);

    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: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
Branch: REC
Department: l 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: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
Branch: REC
Department: l 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 : 30

Section 1 : Coding

1. Problem Statement

Aryan is participating in a coding competition where he needs to sort a list of numbers using an efficient sorting algorithm. He decides to use Merge Sort, a divide-and-conquer algorithm, to achieve this. Given a list of n elements, Aryan must implement merge sort to arrange the numbers in ascending order.

Help Aryan by implementing the merge sort algorithm to correctly sort the given list of numbers.

Input Format

The first line of input contains an integer n, the number of elements in the list. The

second line contains n space-separated integers representing the elements

of the list.

Output Format

The output prints the sorted list of numbers in ascending order, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
80 40 20 50 30
Output: 20 30 40 50 80

Answer

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

void mergesort(int i, int j, int a[], int b[]) { if
   (j <= i)
      return;
   int mid = (i + j) / 2;
   mergesort(i, mid, a, b);
   mergesort(mid + 1, j, a, b); int
   left = i;
   int right = mid + 1;
   int k;
   for (k = i; k <= j; k++) {
      if (left == mid + 1) {
         b[k] = a[right];
         right++;
      } else if (right == j + 1) {
         b[k] = a[left];
         left++;
      } else if (a[left] < a[right]) {
         b[k] = a[left];
         left++;
      } else {
         b[k] = a[right];
         right++;
```

```
        }
    }
    for (k = i; k <= j; k++) {
        a[k] = b[k];
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int* a = (int*)malloc(n * sizeof(int));
    int* b = (int*)malloc(n * sizeof(int)); if
    (!a || !b) {
        printf("Memory allocation error!\n");
        return 1;
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
    mergesort(0, n - 1, a, b);
    for (int i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
    free(a);
    free(b);
    return 0;
}
```

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

2. 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

```
#include <stdio.h>
void evenOddInsertionSort(int arr[], int n) {
   for (int i = 2; i < n; i++) {
      int j = i - 2;
      int temp = arr[i];
```

```
        if ((i + 1) % 2 == 1) {
            while (j >= 0 && temp >= arr[j]) {
                arr[j + 2] = arr[j];
                j -= 2;
            }
            arr[j + 2] = temp;
        } else {
            while (j >= 0 && temp <= arr[j]) {
                arr[j + 2] = arr[j];
                j -= 2;
            }
            arr[j + 2] = temp;
        }
    }
}

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]);
    }

    evenOddInsertionSort(arr, n);
    printArray(arr, 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

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

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];
            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++;
    }
}

bool isPrime(int num) { if
    (num <= 1) {
        return false;
    }
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) {
            return false;
        }
    }
```

```
    return true;
}

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]);
    }

    int primeCount = 0;

    for (int i = 0; i < n; i++) {
        if (isPrime(arr[i])) {
            primeCount++;
        }
    }

     mergeSort(arr, 0, n - 1);

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

    printf("Number of prime integers: %d\n", primeCount); return 0;
}
```

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

Rajalakshmi Engineering College

Name: HEMAPRAKASH KL
Email: 241901036@rajalakshmi.edu.in Roll no: 241901036
Phone: 7010799637
Branch: REC
Department: l 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

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>
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[n1], R[n2];
   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];
         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++;
    }
}

int findDigitSum(int num) { int
    sum = 0;
    while (num > 0) { sum
        += num % 10; num
        /= 10;
    }
    return sum;
}

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]);
    }

    mergeSort(arr, 0, n - 1);
```

```
    int maxDigitSum = 0;
    int maxDigitSumElement = 0;

    for (int i = 0; i < n; i++) {
        int digitSum = findDigitSum(arr[i]); if
        (digitSum > maxDigitSum) {
            maxDigitSum = digitSum;
            maxDigitSumElement = arr[i];
        }
    }

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

    printf("The integer with the highest digit sum is: %d", maxDigitSumElement);

    return 0;
}
```

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

2. 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>
#include <stdlib.h>

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

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, int* iteration) { if
    (low < high) {
        int pivot = partition(arr, low, high);
        (*iteration)++;
        printf("Iteration %d: ", *iteration);
        printArray(arr + low, high - low + 1);
        quickSort(arr, low, pivot - 1, iteration);
        quickSort(arr, pivot + 1, high, iteration);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int* scores = (int*)malloc(n * sizeof(int)); if
    (!scores) {
        printf("Memory allocation error!\n");
        return 1;
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &scores[i]);
    }
    int iteration = 0;
    quickSort(scores, 0, n - 1, &iteration);
    printf("Sorted Order: ");
    printArray(scores, n);
    free(scores);
    return 0;
}
```

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

3. 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>
```

```
#include <stdlib.h>

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[n1], R[n2];

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

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

    int mergedArray[n + m];
    for (int i = 0; i < n; i++) {
        mergedArray[i] = arr1[i];
    }
    for (int i = 0; i < m; i++) {
        mergedArray[n + i] = arr2[i];
    }

    mergeSort(mergedArray, 0, n + m - 1);

    for (int i = 0; i < n + m; i++) {
        if (i > 0 && mergedArray[i] == mergedArray[i - 1]) { continue;
        }
        printf("%d ", mergedArray[i]);
    }

    return 0;
}
```

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

4. 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>
#include <stdlib.h>

int insertionSortSwaps(int arr[], int n) { int
   swaps = 0;

   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 -= 1;
         swaps += 1;
      }

      arr[j + 1] = key;
   }

   return swaps;
}

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

   int* arr = (int*)malloc(n * sizeof(int)); if
   (!arr) {
      printf("Memory allocation error!\n");
      return 1;
   }
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }
   int swaps = insertionSortSwaps(arr, n);
   printf("%d\n", swaps);
   free(arr);
   return 0;
}
```

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

5. 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>

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

    insertionSort(arr, n);

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

    return 0;
}
```

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