

# Rajalakshmi Engineering College

Name: jayanthi v  
Email: 240801131@rajalakshmi.edu.in  
Roll no: 240801131  
Phone: 9487003327  
Branch: REC  
Department: I ECE AE  
Batch: 2028  
Degree: B.E - ECE

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) {
    if (a < b) return -1;
    else if (a > b) return 1;
    else return 0;
}
```

```
void merge(double arr[], int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;
```

```
    double *L = (double *)malloc(n1 * sizeof(double));
    double *R = (double *)malloc(n2 * sizeof(double));
```

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

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

```

    } else {
        arr[k++] = R[j++];
    }
}

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

free(L);
free(R);
}

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: jayanthi v  
Email: 240801131@rajalakshmi.edu.in  
Roll no: 240801131  
Phone: 9487003327  
Branch: REC  
Department: I ECE AE  
Batch: 2028  
Degree: B.E - ECE

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  $n$ th 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  $n$ th 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 - 1; j++) {  
        if (arr[j] < pivot) {  
            i++;  
            int temp = arr[i];  
            arr[i] = arr[j];  
            arr[j] = temp;  
        }  
    }  
    int temp = arr[i + 1];  
    arr[i + 1] = arr[high];  
    arr[high] = temp;  
    return i + 1;  
}
```

```
void quickSort(int arr[], int low, int high) {  
    if (low < high) {  
        int pi = partition(arr, low, high);
```

```

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

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: jayanthi v  
Email: 240801131@rajalakshmi.edu.in  
Roll no: 240801131  
Phone: 9487003327  
Branch: REC  
Department: I ECE AE  
Batch: 2028  
Degree: B.E - ECE

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; 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: jayanthi v  
Email: 240801131@rajalakshmi.edu.in  
Roll no: 240801131  
Phone: 9487003327  
Branch: REC  
Department: I ECE AE  
Batch: 2028  
Degree: B.E - ECE

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) {  
    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;  
    }  
}
```

```
void printArray(int arr[], int n) {  
    for (int i = 0; i < n; i++) {  
        printf("%d", arr[i]);  
        if (i != n - 1) printf(" ");  
    }  
    printf("\n");  
}
```

```
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: jayanthi v  
Email: 240801131@rajalakshmi.edu.in  
Roll no: 240801131  
Phone: 9487003327  
Branch: REC  
Department: I ECE AE  
Batch: 2028  
Degree: B.E - ECE

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++];  
        } else {  
            arr[k++] = right[j++];  
        }  
    }
```

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

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

```

void mergeSort(int arr[], int size) {
    if (size < 2) return;

    int mid = size / 2;
    int left[mid], 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: jayanthi v  
Email: 240801131@rajalakshmi.edu.in  
Roll no: 240801131  
Phone: 9487003327  
Branch: REC  
Department: I ECE AE  
Batch: 2028  
Degree: B.E - ECE

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

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 swap(int* a, int* b) {  
    int temp = *a;  
    *a = *b;  
    *b = temp;  
}
```

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

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

```

    }
}

int main() {
    int n, i;

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

    quickSort(nums, 0, n - 1);

    printf("Sorted colors:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", nums[i]);
    }
    printf("\n");
    return 0;
}

```

**Status :** Correct

**Marks :** 10/10

## 2. 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 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++];
```

```
        else
```

```
            arr[k++] = R[j++];
```

```

    }

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

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

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);
    }
}

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

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

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);

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

printf("Number of prime integers: %d\n", countPrimes(arr, N));

return 0;
}
```

**Status :** Correct

**Marks :** 10/10

### 3. 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>
```

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

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

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

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

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

```
int main() {  
    int n, i;
```

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

quickSort(nums, 0, n - 1);

printf("\n");
for (i = 0; i < n; i++) {
    printf("%d ", nums[i]);
}
printf("\n");
return 0;
}
```

**Status :** Correct

**Marks :** 10/10