

## Rajalakshmi Engineering College

Name: SHARATH BAIRAV  
Email: 240801315@rajalakshmi.edu.in  
Roll no: 240801315  
Phone: 7010809930  
Branch: REC  
Department: I ECE FD  
Batch: 2028  
Degree: B.E - ECE

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 : 1

#### Section 1 : MCQ

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

**Answer**

**Status :** Skipped

**Marks :** 0/1

2. 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**

-

**Status :** Correct

**Marks :** 1/1

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

**Answer**

It as an adaptive sorting algorithm

**Status :** Wrong

**Marks :** 0/1

4. 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

5. 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

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

**Answer**

Two sorted subarrays are combined into one sorted array

**Status :** Correct

**Marks :** 1/1

7. 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 76 67 50

**Status :** Wrong

**Marks :** 0/1

8. Merge sort is \_\_\_\_\_.

**Answer**

Comparison-based sorting algorithm

**Status :** Correct

**Marks :** 1/1

9. Is Merge Sort a stable sorting algorithm?

**Answer**

Yes, always stable.

**Status :** Correct

**Marks :** 1/1

10. 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

11. 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

12. 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**

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

**Answer**

merging

**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. 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**

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

**Answer**

Merge Sort

**Status :** Correct

**Marks :** 1/1

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

**Answer**

When sorting linked lists

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

**Answer**

It requires additional memory for merging

**Status :** Correct

**Marks :** 1/1

20. Which of the following is true about Quicksort?

**Answer**

It is an in-place sorting algorithm

**Status :** Correct

**Marks :** 1/1



# Rajalakshmi Engineering College

Name: SHARATH BAIRAV  
Email: 240801315@rajalakshmi.edu.in  
Roll no: 240801315  
Phone: 7010809930  
Branch: REC  
Department: I ECE FD  
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 : 0

#### 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>
```

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

```
int main() {
```

```
    int N;
```

```
    scanf("%d", &N);
```

```
    int john[N], mary[N];
```

```
    for (int i = 0; i < N; i++)
```

```
        scanf("%d", &john[i]);
```

```
    for (int i = 0; i < N; i++)
```

```
        scanf("%d", &mary[i]);
```

```
    // Reverse Mary's array (descending to ascending)
```

```
    for (int i = 0; i < N / 2; i++) {
```

```
        int temp = mary[i];
```

```
        mary[i] = mary[N - 1 - i];
```

```
        mary[N - 1 - i] = temp;
```

```
    }
```

```
    int i = 0, j = 0, k = 0;
```

```

int merged[2 * N];

while (i < N && j < N) {
    if (john[i] <= mary[j]) {
        merged[k++] = john[i++];
    } else {
        merged[k++] = mary[j++];
    }
}

while (i < N) {
    merged[k++] = john[i++];
}

while (j < N) {
    merged[k++] = mary[j++];
}

for (int x = 0; x < 2 * N; x++) {
    printf("%d ", merged[x]);
}
printf("\n");

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: SHARATH BAIRAV  
Email: 240801315@rajalakshmi.edu.in  
Roll no: 240801315  
Phone: 7010809930  
Branch: REC  
Department: I ECE FD  
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 : 0

#### 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>
```

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

```
int main() {
```

```
    int n;
```

```
    scanf("%d", &n);
```

```
    int arr[n];
```

```
    for(int i = 0; i < n; i++)
```

```
        scanf("%d", &arr[i]);
```

```
    for(int i = 1; i < n; i++) {
```

```
        int key = arr[i];
```

```
        int j = i - 1;
```

```
        // Move elements greater than key to one position ahead
```

```
        while(j >= 0 && arr[j] > key) {
```

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

```
            j--;
```

```
        }
```

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

```
    }
```

```
    for(int i = 0; i < n; i++)
```

```
        printf("%d ", arr[i]);
```

```
    printf("\n");
```

```
    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: SHARATH BAIRAV  
Email: 240801315@rajalakshmi.edu.in  
Roll no: 240801315  
Phone: 7010809930  
Branch: REC  
Department: I ECE FD  
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 : 0

#### 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>
```

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



```

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

```

```

int main() {
    int N;
    scanf("%d", &N);
    char arr[N];
    for (int i = 0; i < N; i++) {
        scanf(" %c", &arr[i]);
    }
    quickSort(arr, 0, N - 1);
    for (int i = 0; i < N; i++) {
        printf("%c ", arr[i]);
    }
    printf("\n");
    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: SHARATH BAIRAV  
Email: 240801315@rajalakshmi.edu.in  
Roll no: 240801315  
Phone: 7010809930  
Branch: REC  
Department: I ECE FD  
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 : 0

#### **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>
```

```
void swap(int *a, int *b) {
```

```
    int temp = *a;
```

```
    *a = *b;
```

```
    *b = temp;
```

```
}
```

```
// Partition function (ascending order)
```

```
int partition(int arr[], int low, int high) {
```

```
    int pivot = arr[low];
```

```
    int i = low + 1;
```

```
    int j = high;
```

```
    while (i <= j) {
```

```
        while (i <= high && arr[i] <= pivot) i++;
```

```
        while (arr[j] > pivot) j--;
```

```
        if (i < j) {
```

```
            swap(&arr[i], &arr[j]);
```

```
        }
```

```
    }
```

```
    swap(&arr[low], &arr[j]);
```

```
    return j;
```

```

}

// QuickSort function
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int p = partition(arr, low, high);
        quickSort(arr, low, p - 1);
        quickSort(arr, p + 1, high);
    }
}

// Find k-th largest element
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: SHARATH BAIRAV  
Email: 240801315@rajalakshmi.edu.in  
Roll no: 240801315  
Phone: 7010809930  
Branch: REC  
Department: I ECE FD  
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 : 0

#### 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;
}

void merge(double arr[], int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;
    double left[n1], right[n2];
    for (int i = 0; i < n1; i++) left[i] = arr[l + i];
    for (int j = 0; j < n2; j++) right[j] = arr[m + 1 + j];
    int i = 0, j = 0, k = l;
    while (i < n1 && j < n2) {
        if (compare(left[i], right[j])) {
            arr[k++] = left[i++];
        } else {
            arr[k++] = right[j++];
        }
    }
    while (i < n1) arr[k++] = left[i++];
    while (j < n2) arr[k++] = right[j++];
}

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