LPU_Colab

admin.lpucolab438.examly.io/course

DSA_Unit 4_Lecture 23 Coding
Test Summary

- No. of Sections: 1
- No. of Questions: 12
- Total Duration: 90 min

Section 1 - Coding
Section Summary

- No. of Questions: 12
- Duration: 90 min

Additional Instructions:

None
Q1.
Problem Statement

Alex is a young computer science enthusiast who loves solving coding problems. One day, Alex stumbled upon a unique challenge related to sorting numbers. The challenge involves sorting a list of integers with a unique twist: Alex aims to place the even numbers in their original input order first, followed by the odd numbers in the same input order.

Your task is to help Alex implement a logic of merge sort and a recursive function to arrange the even and odd numbers separately.

Input Format
The first line contains an integer n, the number of integers in the list.

The second line contains n space-separated integers, a[i], representing the elements of the list.

Output Format
The output displays a single line containing n space-separated integers, representing the sorted list, with even numbers appearing first in the given input order, followed by odd numbers in the given input order.

Refer to the sample outputs for the exact format.

Constraints
1 ≤ n ≤ 30

-100 <= a[i] <= 100

Sample Input Sample Output

```
4
1 2 3 4

2 4 1 3
```

Sample Input Sample Output

```
5
1 5 3 -4 0

-4 0 1 5 3
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Q2.
Problem Statement

You have a dictionary with a collection of characters. Implement a program that sorts the characters in lexicographic order using the recursive Merge Sort algorithm.

Call the function **mergeSort** to sort the array of characters. The program should take the unsorted list of characters as input and output the sorted list of characters.

Write a logic for merge sorting and a recursive function.

Note: The lexicographical order of characters will be 'A', 'B', 'C', …, 'Y', 'Z', 'a', 'b', 'c', …, 'y', 'z'.

Input Format
The first line of input consists of the characters n, representing the number of charactersin the dictionary.

The second line of input consists of **n** characters, separated by spaces.

Output Format

The output displays a sorted list of characters in lexicographic order, with each character separated by a space.

Refer to the sample output for the formatting specifications.

Constraints
MAX_Char_Length 30

1 ≤ n ≤ 30

Input characters should be lowercase or uppercase.

Sample Input Sample Output

```
4
z g v a

Sorted Characters: a g v z
```

Sample Input Sample Output

```
3
a b a

Sorted Characters: a a b
```

Sample Input Sample Output

```
5
a b A c D

Sorted Characters: A D a b c
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Q3.
Problem Statement

Arsha is working on a project where binary data is an essential part of the processing. To optimize data handling, Arsha needs to sort an array containing only two types of elements: 0s and 1s. Alex decides to implement a merge sort algorithm to efficiently sort this binary data.

Write a program to assist Arsha in implementing the logic of merge sort along with a recursive function to sort an array of binary data in ascending order.

Input Format

The first line contains an integer, representing the number of elements in the array.

The second line contains n space-separated integers, where each integer is either 0 or 1.

Output Format
The output displays the following result:

- If the input contains elements other than 0 and 1, print "Invalid input.".
- Otherwise, print a single line containing n space-separated integers, representing the sorted array in ascending order.

Refer to the sample outputs for the exact format.

Constraints
1 ≤ n ≤ 20

input = 0, and 1 only

Sample Input Sample Output

```
5
1 0 1 0 1

0 0 1 1 1
```

Sample Input Sample Output

```
3
1 0 2

Invalid input.
```

Sample Input Sample Output

```
4
1 1 1 1

1 1 1 1
```

Sample Input Sample Output

```
2
0 0

0 0
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Q4.
Problem Statement

You are developing a leaderboard system for an online game. The scores of the players are stored in an array of integers. To display the leaderboard, you need to sort the scores in ascending order.

Write code to implement a recursive **merge sort** algorithm for sorting the scores.

Note: This kind of question will help in clearing Wipro recruitment.

Input Format
The first line of input consists of an integer n, representing the number of scores in the array.

The second line of input consists of **n** space-separated integers, representing the scores of the players.

Output Format
The first line of output displays the initial array of scores before sorting.

The second line of output displays the sorted array of scores after applying the merge sort algorithm.

Refer to the sample output for the formatting specifications.

Constraints
1 <= n <= 10

0 <= scores <= 100

Sample Input Sample Output

```
5
3 1 4 2 5

3 1 4 2 5
1 2 3 4 5
```

Sample Input Sample Output

```
3
9 7 8

9 7 8
7 8 9
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Q5.
Problem Statement

In a mystical land known as Eldoria, ancient wizards use magical runes to cast powerful spells. These runes are represented by single characters, each possessing a unique magical property. However, the wizards have a challenge: they need these magical runes sorted in a specific order for their spells to work correctly.

Write a program to help the wizards of Eldoria sort their magical runes based on their potency. Each rune is represented by a single character, and each character holds a unique level of magical power, determined by its position in the ASCII table.

Your task is to implement a merge sorting logic and recursive function to arrange these magical runes in descending order of their magical potency.

Input Format
The first line of input consists of an integer n, representing the number of magical runes to be sorted.

The second line contains n space-separated characters, each representing a magical rune.

Output Format
The output displays a single line containing the magical runes sorted in descending order of their magical potency, separated by spaces.

Refer to the sample output for the formatting specifications.

Constraints
1 <= n <= 25

Each character in the array is a single uppercase letter or a single lowercase letter.

Sample Input Sample Output

```
8
G h I J K L m n

n m h L K J I G
```

Sample Input Sample Output

```
6
a Z A B M Z

a Z Z M B A
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Q6.
Problem Statement

At the "Fresh Mart" grocery store, they recently received a large shipment of various products, and they need to organize their inventory efficiently. They've hired you to help them sort the products in descending order based on their product IDs.

Develop a program to aid the Fresh Mart team in organizing their inventory using a recursive function and implementing the merge sort logic. Each product in the store is identified by a distinct product ID, and the objective is to present the products to customers in descending order.

Input Format
The first line consists of an integer, n, representing the number of products in the inventory.

The next line consists of n space-separated integers, each representing a unique Product ID.

Output Format
The output displays a single line containing the Product IDs sorted in descending order, separated by spaces.

Constraints
1 ≤ n ≤ 10

1 ≤ Product ID ≤ 1000

Sample Input Sample Output

```
5
1 9 7 6 8

9 8 7 6 1
```

Sample Input Sample Output

```
4
754 312 505 44

754 505 312 44
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Q7.
Problem Statement

Alice is working on a project that requires sorting and displaying the frequency of elements in an array. She needs your help to write a program to sort an array of positive integers and display the elements sorted by frequency in descending order. If two elements have the same frequency, they should be sorted in ascending order of their values.

For this project, your task is to implement the logic of the merge sort and a recursive function.

Input Format
The first line contains an integer n, representing the number of elements in the array.

The second line contains n space-separated integers, arr[i].

Output Format
The output prints a single line containing the sorted elements of the array separated by spaces.

Constraints
max_n = 100

1 ≤ n ≤ 25

1 <= arr[i] <= 100

Sample Input Sample Output

```
6
1 1 2 3 3 3

3 3 3 1 1 2
```

Sample Input Sample Output

```
7
2 2 3 1 3 2 3

2 2 2 3 3 3 1
```

Sample Input Sample Output

```
6
1 2 3 1 2 3
```

```
1 1 2 2 3 3
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Q8.
Problem Statement

John is a linguistic enthusiast who is fascinated by the arrangement of letters based on their characteristics. He is particularly interested in sorting characters based on whether they are vowels or consonants.

Write a program to sort a given set of characters. The characters should be sorted in such a way that vowels appear first, followed by consonants. Within the vowels and consonants, maintain the original input order of characters.

Your task is to assist John in implementing the logic for merge sort and a recursive function to achieve the goal stated in the program.

Input Format
The first line consists of an integer n, representing the number of characters in the list.

The next line consists of n space-separated characters, containing alphabetic characters (both uppercase and lowercase).

Output Format
The output displays the characters sorted first by vowels, maintaining the original order of input, followed by consonants in the same input order.

Constraints
1 ≤ n ≤ 15

1 ≤ |characters| ≤ 15

input characters should be lowercase or uppercase letters

Sample Input Sample Output

```
8
e A b a i o u D
```

```
e A a i o u b D
```

Sample Input Sample Output

```
8
r a E y T m b N
```

```
a E r y T m b N
```

Sample Input Sample Output

```
3
b h l

b h l
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Q9.
Problem Statement

Sarah is an avid programmer who loves to solve interesting problems. Today, she encountered a unique challenge related to sorting odd numbers from an array of integers. She wants to sort the odd numbers in ascending order while keeping the even numbers in their original positions.

Sarah has contacted you for help with this challenge. Can you implement both the logic of merge sort and a recursive function to achieve the above task?

Input Format
The first line contains an integer n, representing the number of elements in the array.

The second line contains n space-separated integers, representing the elements of the array.

Output Format
If there are no odd numbers in the array, output prints "None".

If there are odd numbers in the array, the output displays sorted odd numbers separated by a space in ascending order.

Refer to the sample output for the formatting specifications.

Constraints
$1 \leq n \leq 10$

$-10^3 \leq arr[i] \leq 10^3$

Sample Input Sample Output

```
5
105 236 789 415 364
```

```
105 415 789
```

Sample Input Sample Output

```
4
-9 -47 56 32

-47 -9
```

Sample Input Sample Output

```
3
2 4 6

None
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Q10.
Problem Statement

You are a student attending a class on sorting techniques taught by Professor Smith. Today, Professor Smith is explaining how to efficiently sort both positive and negative values using the merge sort algorithm and a recursive function. To better understand the concept, you decide to write a program.

Your task is to write code to implement a recursive merge sort algorithm for sorting positive and negative floating-point numbers, as explained by Professor Smith.

Input Format
The first line contains an integer n, representing the number of values to be sorted.

The second line contains n space-separated floating-point numbers.

Output Format
The output displays a single line containing n space-separated floating-point numbers, representing the sorted values in ascending order. Each number is rounded to two decimal places.

Refer to the sample output for formatting specifications.

Constraints
1 ≤ n ≤ 15

-100.0 ≤ floating point numbers ≤ 100.0

Sample Input Sample Output

```
5
3.14 1.1 2.71 0.5 1.618

Sorted Array:
0.50 1.10 1.62 2.71 3.14
```

Sample Input Sample Output

```
6
-2.5 5.0 0.0 -1.5 2.0 3.5

Sorted Array:
-2.50 -1.50 0.00 2.00 3.50 5.00
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Q11.
Problem Statement

John is a student who just received his test scores (floating-point numbers) for various subjects. He wants to organize his scores in descending order so that he can see his highest scores first. Can you help him write a program to sort his test scores in descending order?

Your task is to write a program to sort John's test scores in descending order using the Merge Sort algorithm and a recursive function.

Input Format
The first line contains an integer, n, representing the number of test scores John has received.

The next line contains n floating-point numbers separated by spaces, each representing John's test score.

Output Format
The output displays the sorted test scores in descending order, each rounded to two decimal places.

Refer to the sample outputs for the exact format.

Constraints
1 ≤ n ≤ 15

0.0 <= test score <= 100.0

Sample Input Sample Output

```
4
9.8 1.1 3.3 7.7

Sorted Array:
9.80 7.70 3.30 1.10
```

Sample Input Sample Output

```
5
3.14 1.1 2.71 0.5 1.618

Sorted Array:
3.14 2.71 1.62 1.10 0.50
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Q12.
Problem Statement

Raj wants to learn a sequence of integers. His task is to count the frequency of each unique integer in the sequence and then sort them based on their frequencies in ascending order. If two integers have the same frequency, sort them in ascending order of their values.

For this goal, your task is to implement the logic of the merge sort and a recursive function.

Example 1

Input:

nums = [1, 1, 2, 2, 2, 3]

Output:

[3,1,1,2,2,2]

Explanation:

'3' has a frequency of 1, '1' has a frequency of 2, and '2' has a frequency of 3.

Example 2

Input:

nums = [3, 2, 1, 3, 2]

Output:

[1,2,2,3,3]

Explanation:

'3' and '2' both have a frequency of 2, so they are sorted based on their actual values, with the smaller value coming first.

Input Format
The first line of input contains an integer, n, representing the number of integers in the sequence.

The second line of input contains n space-separated integers, arr [i], representing the elements of the sequence.

Output Format
The output displays a single line containing the sorted integers separated by space according to their frequencies and values.

Constraints
max_n = 100

1 <= n <= 25

1 <= arr[i] <= 100

Sample Input Sample Output

```
6
1 1 2 2 2 3
```

```
3 1 1 2 2 2
```

Sample Input Sample Output

```
5
3 2 1 3 2

1 2 2 3 3
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb
Answer Key & Solution

Section 1 - Coding
Q1 Test Case Input Output

```
4
9 2 7 4

2 4 9 7
```

Weightage - 10 Input Output

```
6
12 45 7 24 9 17

12 24 45 7 9 17
```

Weightage - 15 Input Output

```
4
100 17 41 120

100 120 17 41
```

Weightage - 15 Input Output

```
3
5 10 30

10 30 5
```

Weightage - 10 Input Output

```
10
-2 3 -4 5 -6 7 -8 9 11 3

-2 -4 -6 -8 3 5 7 9 11 3
```

Weightage - 25 Input Output

```
30
50 10 90 30 80 20 70 60 20 34 2 3 4 5 6 2 3 4 5 6 3 5 2 7 8 9 11 23 24 56

50 10 90 30 80 20 70 60 20 34 2 4 6 2 4 6 2 8 24 56 3 5 3 5 3 5 7 9 11 23
```

Weightage - 25 Sample Input Sample Output

```
4
1 2 3 4
```

```
2 4 1 3
```

Sample Input Sample Output

```
5
1 5 3 -4 0

-4 0 1 5 3
```

Solution

```
#include <iostream>

using namespace std;

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 j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

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

    while (i < n1 && (L[i] % 2 == 0)) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2 && (R[j] % 2 == 0)) {
        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;
    cin >> n;
```

```
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    mergeSort(arr, 0, n - 1);

    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

```c
#include <stdio.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 j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

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

    while (i < n1 && (L[i] % 2 == 0)) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2 && (R[j] % 2 == 0)) {
        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;
    scanf("%d", &n);

    int arr[n];
    for (int i = 0; i < n; i++) {
```

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

    mergeSort(arr, 0, n - 1);

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

    return 0;
}
```

Q2 Test Case Input Output

```
6
w x y z a b

Sorted Characters: a b w x y z
```

Weightage - 10 Input Output

```
5
q p o n m

Sorted Characters: m n o p q
```

Weightage - 10 Input Output

```
10
e j k p l n m i h o

Sorted Characters: e h i j k l m n o p
```

Weightage - 15 Input Output

```
12
c a d b f e h j i g k l

Sorted Characters: a b c d e f g h i j k l
```

Weightage - 15 Input Output

```
15
e d C b a f g h i j K l m N o

Sorted Characters: C K N a b d e f g h i j l m o
```

Weightage - 25 Input Output

```
26
g p j h a l e c f o b n m d k i r q s t u v w x y z

Sorted Characters: a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Weightage - 25 Sample Input Sample Output

```
4
z g v a

Sorted Characters: a g v z
```

Sample Input Sample Output

```
3
a b a

Sorted Characters: a a b
```

Sample Input Sample Output

```
5
a b A c D

Sorted Characters: A D a b c
```

Solution

```
#include <stdio.h>

#define MAX_Char_LENGTH 30

void merge(char arr[][MAX_Char_LENGTH], int left, int mid, int right) {
    int i = left;
    int j = mid + 1;
    int k = 0;
    char temp[right - left + 1][MAX_Char_LENGTH];

    while (i <= mid && j <= right) {
        if (arr[i][0] <= arr[j][0]) {
            int l = 0;
            while (arr[i][l] != '\0') {
                temp[k][l] = arr[i][l];
                l++;
            }
            temp[k][l] = '\0';
            i++;
        } else {
            int l = 0;
            while (arr[j][l] != '\0') {
                temp[k][l] = arr[j][l];
                l++;
            }
            temp[k][l] = '\0';
            j++;
        }
        k++;
    }

    while (i <= mid) {
        int l = 0;
        while (arr[i][l] != '\0') {
            temp[k][l] = arr[i][l];
            l++;
        }
        temp[k][l] = '\0';
        i++;
        k++;
    }

    while (j <= right) {
        int l = 0;
        while (arr[j][l] != '\0') {
            temp[k][l] = arr[j][l];
            l++;
        }
        temp[k][l] = '\0';
        j++;
        k++;
    }

    for (int p = left, q = 0; p <= right; p++, q++) {
        int l = 0;
        while (temp[q][l] != '\0') {
```

```
            arr[p][l] = temp[q][l];
            l++;
        }
        arr[p][l] = '\0';
    }
}

void mergeSort(char arr[][MAX_Char_LENGTH], 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 size;
    scanf("%d", &size);

    char arr[size][MAX_Char_LENGTH];

    for (int i = 0; i < size; i++) {
        scanf("%s", arr[i]);
    }

    mergeSort(arr, 0, size - 1);

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

    return 0;
}
```

```
#include <iostream>

#define MAX_Char_LENGTH 30

using namespace std;

void merge(char arr[][MAX_Char_LENGTH], int left, int mid, int right) {
    int i = left;
    int j = mid + 1;
    int k = 0;
    char temp[right - left + 1][MAX_Char_LENGTH];

    while (i <= mid && j <= right) {
        if (arr[i][0] <= arr[j][0]) {
            temp[k][0] = arr[i][0];
            i++;
        } else {
            temp[k][0] = arr[j][0];
            j++;
        }
        k++;
    }

    while (i <= mid) {
        temp[k][0] = arr[i][0];
        i++;
        k++;
    }

    while (j <= right) {
        temp[k][0] = arr[j][0];
        j++;
        k++;
    }

    for (int p = left, q = 0; p <= right; p++, q++) {
        arr[p][0] = temp[q][0];
    }
}

void mergeSort(char arr[][MAX_Char_LENGTH], 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 size;
    cin >> size;

    char arr[size][MAX_Char_LENGTH];
```

```
    for (int i = 0; i < size; i++) {
        cin >> arr[i];
    }

    mergeSort(arr, 0, size - 1);

    cout << "Sorted Characters: ";
    for (int i = 0; i < size; i++) {
        cout << arr[i][0] << " ";
    }
    cout << "\n";

    return 0;
}
```

Q3 Test Case Input Output

```
7
1 0 1 0 1 0 2

Invalid input.
```

Weightage - 10 Input Output

```
10
0 1 0 0 1 1 0 1 0 1

0 0 0 0 0 1 1 1 1 1
```

Weightage - 10 Input Output

```
12
0 1 0 1 0 1 0 1 0 1 0 1

0 0 0 0 0 0 1 1 1 1 1 1
```

Weightage - 15 Input Output

```
10
0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0
```

Weightage - 15 Input Output

```
15
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Weightage - 25 Input Output

```
20
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
```

Weightage - 25 Sample Input Sample Output

```
5
1 0 1 0 1

0 0 1 1 1
```

Sample Input Sample Output

```
3
1 0 2

Invalid input.
```

Sample Input Sample Output

```
4
1 1 1 1

1 1 1 1
```

Sample Input Sample Output

```
2
0 0

0 0
```

Solution

```c
#include <stdio.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 j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    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++;
    }
}

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]);
        if (arr[i] != 0 && arr[i] != 1) {
            printf("Invalid input.\n");
            return 0;
        }
    }

    mergeSort(arr, 0, n - 1);

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

    return 0;
}
```

```
#include <iostream>
using namespace std;

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 j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    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++;
    }
}

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;
    cin >> n;

    int arr[n];
```

```
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
        if (arr[i] != 0 && arr[i] != 1) {
            cout << "Invalid input." << endl;
            return 0;
        }
    }

    mergeSort(arr, 0, n - 1);

    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Q4 Test Case Input Output

```
8
5 3 7 1 8 2 6 4

5 3 7 1 8 2 6 4
1 2 3 4 5 6 7 8
```

Weightage - 20 Input Output

```
5
27 19 36 14 5

27 19 36 14 5
5 14 19 27 36
```

Weightage - 15 Input Output

```
7
42 38 51 47 56 40 33

42 38 51 47 56 40 33
33 38 40 42 47 51 56
```

Weightage - 15 Input Output

```
7
25 17 20 23 18 16 22

25 17 20 23 18 16 22
16 17 18 20 22 23 25
```

Weightage - 15 Input Output

```
10
23 61 85 96 74 56 85 14 20 0
```

```
23 61 85 96 74 56 85 14 20 0
0 14 20 23 56 61 74 85 85 96
```

Weightage - 20 Input Output

```
7
18 22 15 10 24 19 13

18 22 15 10 24 19 13
10 13 15 18 19 22 24
```

Weightage - 15 Sample Input Sample Output

```
5
3 1 4 2 5

3 1 4 2 5
1 2 3 4 5
```

Sample Input Sample Output

```
3
9 7 8

9 7 8
7 8 9
```

Solution

```c
#include <stdio.h>

void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    int leftArr[n1];
    int rightArr[n2];

    for (int i = 0; i < n1; i++) {
        leftArr[i] = arr[left + i];
    }
    for (int i = 0; i < n2; i++) {
        rightArr[i] = arr[mid + 1 + i];
    }

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

    while (i < n1 && j < n2) {
        if (leftArr[i] <= rightArr[j]) {
            arr[k] = leftArr[i];
            i++;
        } else {
            arr[k] = rightArr[j];
            j++;
        }
        k++;
    }

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

    while (j < n2) {
        arr[k] = rightArr[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 size;
    scanf("%d", &size);
```

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

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

    mergeSort(array, 0, size - 1);

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

    return 0;
}
```

```
#include <iostream>
using namespace std;

void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    int leftArr[n1];
    int rightArr[n2];

    for (int i = 0; i < n1; i++) {
        leftArr[i] = arr[left + i];
    }
    for (int i = 0; i < n2; i++) {
        rightArr[i] = arr[mid + 1 + i];
    }

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

    while (i < n1 && j < n2) {
        if (leftArr[i] <= rightArr[j]) {
            arr[k] = leftArr[i];
            i++;
        } else {
            arr[k] = rightArr[j];
            j++;
        }
        k++;
    }

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

    while (j < n2) {
        arr[k] = rightArr[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 size;
```

```
    cin >> size;

    int array[size];
    for (int i = 0; i < size; i++) {
        cin >> array[i];
    }

    for (int i = 0; i < size; i++) {
        cout << array[i] << " ";
    }
    cout << endl;

    mergeSort(array, 0, size - 1);

    for (int i = 0; i < size; i++) {
        cout << array[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Q5 Test Case Input Output

```
7
a C d B E E G

d a G E E C B
```

Weightage - 10 Input Output

```
5
a b c d i

i d c b a
```

Weightage - 10 Input Output

```
10
F B K G j C l D A I

l j K I G F D C B A
```

Weightage - 15 Input Output

```
12
L K J I H g F E D c B a

g c a L K J I H F E D B
```

Weightage - 15 Input Output

```
24
X W V U t S R q P O N M L k J I H g F E D C B A

t q k g X W V U S R P O N M L J I H F E D C B A
```

Weightage - 25 Input Output

```
25
P q R s T U V W X Y Z A b C D e F G H I j K L m N O

s q m j e b Z Y X W V U T R P N L K I H G F D C A
```

Weightage - 25 Sample Input Sample Output

```
8
G h I J K L m n

n m h L K J I G
```

Sample Input Sample Output

```
6
a Z A B M Z

a Z Z M B A
```

Solution

```c
#include <stdio.h>

#define MAX_ARRAY_LENGTH 100 // Maximum array length

void merge_descending(char arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    char left_arr[n1];
    char right_arr[n2];

    for (int i = 0; i < n1; i++)
        left_arr[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        right_arr[j] = arr[mid + 1 + j];

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (left_arr[i] >= right_arr[j]) {  // Compare in descending order
            arr[k] = left_arr[i];
            i++;
        } else {
            arr[k] = right_arr[j];
            j++;
        }
        k++;
    }

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

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

void mergeSortDescending(char arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSortDescending(arr, left, mid);
        mergeSortDescending(arr, mid + 1, right);
        merge_descending(arr, left, mid, right);
    }
}

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

    char arr[n];
    for (int i = 0; i < n; i++)
```

```
        scanf(" %c", &arr[i]); // Read single characters

    mergeSortDescending(arr, 0, n - 1);

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

    return 0;
}
```

```
#include <iostream>
using namespace std;

const int MAX_ARRAY_LENGTH = 100; // Maximum array length

void merge_descending(char arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    char left_arr[n1];
    char right_arr[n2];

    for (int i = 0; i < n1; i++)
        left_arr[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        right_arr[j] = arr[mid + 1 + j];

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (left_arr[i] >= right_arr[j]) {  // Compare in descending order
            arr[k] = left_arr[i];
            i++;
        } else {
            arr[k] = right_arr[j];
            j++;
        }
        k++;
    }

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

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

void mergeSortDescending(char arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSortDescending(arr, left, mid);
        mergeSortDescending(arr, mid + 1, right);
        merge_descending(arr, left, mid, right);
    }
}

int main() {
    int n;
    cin >> n;

    char arr[MAX_ARRAY_LENGTH];
```

```
    for (int i = 0; i < n; i++)
        cin >> arr[i]; // Read single characters

    mergeSortDescending(arr, 0, n - 1);

    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";

    return 0;
}
```

Q6 Test Case Input Output

```
5
1 7 98 6 3

98 7 6 3 1
```

Weightage - 10 Input Output

```
5
78 6 4 2 3

78 6 4 3 2
```

Weightage - 10 Input Output

```
7
4 5 9 7 1 3 6

9 7 6 5 4 3 1
```

Weightage - 15 Input Output

```
7
9 6 3 1 4 8 2

9 8 6 4 3 2 1
```

Weightage - 15 Input Output

```
10
117 216 43 114 578 125 182 876 710 912

912 876 710 578 216 182 125 117 114 43
```

Weightage - 25 Input Output

```
10
7 9 63 45 1 8 3 4 91 22

91 63 45 22 9 8 7 4 3 1
```

Weightage - 25 Sample Input Sample Output

```
5
1 9 7 6 8

9 8 7 6 1
```

Sample Input Sample Output

```
4
754 312 505 44

754 505 312 44
```

Solution

```
#include <iostream>

using namespace std;

// A function to merge the two half into a sorted data.
void Merge(int *a, int low, int high, int mid)
{
        // We have low to mid and mid+1 to high already sorted.
        int i, j, k, temp[high-low+1];
        i = low;
        k = 0;
        j = mid + 1;

        // Merge the two parts into temp[].
        while (i <= mid && j <= high)
        {
                if (a[i] < a[j])
                {
                        temp[k] = a[i];
                        k++;
                        i++;
                }
                else
                {
                        temp[k] = a[j];
                        k++;
                        j++;
                }
        }

        // Insert all the remaining values from i to mid into temp[].
        while (i <= mid)
        {
                temp[k] = a[i];
                k++;
                i++;
        }

        // Insert all the remaining values from j to high into temp[].
        while (j <= high)
        {
                temp[k] = a[j];
                k++;
                j++;
        }


        // Assign sorted data stored in temp[] to a[].
        for (i = low; i <= high; i++)
        {
                a[i] = temp[i-low];
        }
}

// A function to split array into two parts.
void MergeSort(int *a, int low, int high)
```

```
{
        int mid;
        if (low < high)
        {
                mid=(low+high)/2;
                // Split the data into two half.
                MergeSort(a, low, mid);
                MergeSort(a, mid+1, high);

                // Merge them to get sorted output.
                Merge(a, low, high, mid);
        }
}

int main()
{
        int n, i;
        cin>>n;

        int arr[n];
        for(i = 0; i < n; i++)
        {
                cin>>arr[i];
        }

        MergeSort(arr, 0, n-1);
        for (i = (n-1); i >=0; i--)
        cout<<arr[i]<<" ";

        return 0;
}
```

```
#include <stdio.h>

// A function to merge the two halves into a sorted data.
void Merge(int *a, int low, int high, int mid) {
    int i, j, k, temp[high - low + 1];
    i = low;
    k = 0;
    j = mid + 1;

    // Merge the two parts into temp[].
    while (i <= mid && j <= high) {
        if (a[i] < a[j]) {
            temp[k] = a[i];
            k++;
            i++;
        } else {
            temp[k] = a[j];
            k++;
            j++;
        }
    }

    // Insert all the remaining values from i to mid into temp[].
    while (i <= mid) {
        temp[k] = a[i];
        k++;
        i++;
    }

    // Insert all the remaining values from j to high into temp[].
    while (j <= high) {
        temp[k] = a[j];
        k++;
        j++;
    }

    // Assign sorted data stored in temp[] to a[].
    for (i = low; i <= high; i++) {
        a[i] = temp[i - low];
    }
}

// A function to split the array into two parts.
void MergeSort(int *a, int low, int high) {
    int mid;
    if (low < high) {
        mid = (low + high) / 2;
        // Split the data into two halves.
        MergeSort(a, low, mid);
        MergeSort(a, mid + 1, high);

        // Merge them to get sorted output.
        Merge(a, low, high, mid);
    }
}
```

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

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

    MergeSort(arr, 0, n - 1);
    for (i = (n - 1); i >= 0; i--) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Q7 Test Case Input Output

```
9
1 1 6 4 5 6 1 4 1

1 1 1 1 4 4 6 6 5
```

Weightage - 15 Input Output

```
12
6 56 15 45 74 87 96 5 4 2 3 56

56 56 2 3 4 5 6 15 45 74 87 96
```

Weightage - 15 Input Output

```
15
5 98 85 84 21 45 5 5 78 5 63 21 98 78 21

5 5 5 5 21 21 21 78 78 98 98 45 63 84 85
```

Weightage - 25 Input Output

```
25
89 6 9 8 7 55 55 78 5 5 78 5 78 89 6 9 45 65 79 78 45 65 78 89 9

78 78 78 78 78 5 5 5 9 9 9 89 89 89 6 6 45 45 55 55 65 65 7 8 79
```

Weightage - 25 Input Output

```
8
3 2 5 3 5 2 1 5

5 5 5 2 2 3 3 1
```

Weightage - 10 Input Output

```
7
3 3 8 9 1 1 1
```

```
1 1 1 3 3 8 9
```

Weightage - 10 Sample Input Sample Output

```
6
1 1 2 3 3 3

3 3 3 1 1 2
```

Sample Input Sample Output

```
7
2 2 3 1 3 2 3

2 2 2 3 3 3 1
```

Sample Input Sample Output

```
6
1 2 3 1 2 3

1 1 2 2 3 3
```

Solution

```
#include <stdio.h>

const int MAX_N = 100;

void merge(int arr[], int temp[], 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 j = 0; j < n2; j++) R[j] = arr[mid + 1 + j];

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

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

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

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

        mergeSort(arr, temp, left, mid);
        mergeSort(arr, temp, mid + 1, right);
        merge(arr, temp, left, mid, right);
    }
}

void countAndSortFrequency(int arr[], int n) {
    int freqArr[MAX_N][2] = {0};
    int values[MAX_N];

    int index = 0;
    for (int i = 0; i < n; i++) {
        int isAlreadyCounted = 0;
        for (int j = 0; j < index; j++) {
            if (values[j] == arr[i]) {
                freqArr[j][1]++;
                isAlreadyCounted = 1;
                break;
            }
        }
        if (!isAlreadyCounted) {
            values[index] = arr[i];
            freqArr[index][0] = arr[i];
            freqArr[index][1] = 1;
            index++;
```

```
        }
    }

    for (int i = 0; i < index; i++) {
        for (int j = i + 1; j < index; j++) {
            if (freqArr[i][1] < freqArr[j][1] || (freqArr[i][1] == freqArr[j][1]
&& freqArr[i][0] > freqArr[j][0])) {
                int temp = freqArr[i][0];
                freqArr[i][0] = freqArr[j][0];
                freqArr[j][0] = temp;

                temp = freqArr[i][1];
                freqArr[i][1] = freqArr[j][1];
                freqArr[j][1] = temp;
            }
        }
    }

    for (int i = 0; i < index; i++) {
        for (int j = 0; j < freqArr[i][1]; j++) {
            printf("%d ", freqArr[i][0]);
        }
    }
    printf("\n");
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[MAX_N];

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

    countAndSortFrequency(arr, n);

    return 0;
}
```

```
#include <iostream>

using namespace std;

const int MAX_N = 100;

void merge(int arr[], int temp[], 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 j = 0; j < n2; j++) R[j] = arr[mid + 1 + j];

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

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

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

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

        mergeSort(arr, temp, left, mid);
        mergeSort(arr, temp, mid + 1, right);
        merge(arr, temp, left, mid, right);
    }
}

void countAndSortFrequency(int arr[], int n) {
    int freqArr[MAX_N][2] = {0};
    int values[MAX_N];

    int index = 0;
    for (int i = 0; i < n; i++) {
        int isAlreadyCounted = 0;
        for (int j = 0; j < index; j++) {
            if (values[j] == arr[i]) {
                freqArr[j][1]++;
                isAlreadyCounted = 1;
                break;
            }
        }
        if (!isAlreadyCounted) {
            values[index] = arr[i];
            freqArr[index][0] = arr[i];
```

```
            freqArr[index][1] = 1;
            index++;
        }
    }

    for (int i = 0; i < index; i++) {
        for (int j = i + 1; j < index; j++) {
            if (freqArr[i][1] < freqArr[j][1] || (freqArr[i][1] == freqArr[j][1]
&& freqArr[i][0] > freqArr[j][0])) {
                int temp = freqArr[i][0];
                freqArr[i][0] = freqArr[j][0];
                freqArr[j][0] = temp;

                temp = freqArr[i][1];
                freqArr[i][1] = freqArr[j][1];
                freqArr[j][1] = temp;
            }
        }
    }

    for (int i = 0; i < index; i++) {
        for (int j = 0; j < freqArr[i][1]; j++) {
            cout << freqArr[i][0] << " ";
        }
    }
    cout << endl;
}

int main() {
    int n;
    cin >> n;
    int arr[MAX_N];

    for (int i = 0; i < n; i++) cin >> arr[i];

    countAndSortFrequency(arr, n);

    return 0;
}
```

Q8 Test Case Input Output

```
3
y X Z

y X Z
```

Weightage - 10 Input Output

```
7
J i h I j G H

i I J h j G H
```

Weightage - 15 Input Output

```
5
p O q R s

O p q R s
```

Weightage - 10 Input Output

```
10
P p C c F f Z z R r

P p C c F f Z z R r
```

Weightage - 15 Input Output

```
12
L g H t a X b Z R Y p u

a u L g H t X b Z R Y p
```

Weightage - 25 Input Output

```
15
X o L e O y O U b e l i e v e

o e O O U e i e e X L y b l v
```

Weightage - 25 Sample Input Sample Output

```
8
e A b a i o u D

e A a i o u b D
```

Sample Input Sample Output

```
8
r a E y T m b N

a E r y T m b N
```

Sample Input Sample Output

```
3
b h l

b h l
```

Solution

```
#include <iostream>

using namespace std;
const int MAX_N = 100;

int isConsonant(char ch) {
    if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {
        if (ch != 'a' && ch != 'e' && ch != 'i' && ch != 'o' && ch != 'u' &&
            ch != 'A' && ch != 'E' && ch != 'I' && ch != 'O' && ch != 'U') {
            return 1;
        }
    }
    return 0;
}

void merge(char characters[], int left, int middle, int right, int
consonant_count[]) {
    int n1 = middle - left + 1;
    int n2 = right - middle;

    char leftArray[MAX_N];
    char rightArray[MAX_N];
    int leftConsonantCount[MAX_N];
    int rightConsonantCount[MAX_N];

    for (int i = 0; i < n1; i++) {
        leftArray[i] = characters[left + i];
        leftConsonantCount[i] = consonant_count[left + i];
    }
    for (int j = 0; j < n2; j++) {
        rightArray[j] = characters[middle + 1 + j];
        rightConsonantCount[j] = consonant_count[middle + 1 + j];
    }

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (leftConsonantCount[i] <= rightConsonantCount[j]) {
            characters[k] = leftArray[i];
            consonant_count[k] = leftConsonantCount[i];
            i++;
        } else {
            characters[k] = rightArray[j];
            consonant_count[k] = rightConsonantCount[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        characters[k] = leftArray[i];
        consonant_count[k] = leftConsonantCount[i];
        i++;
        k++;
    }

    while (j < n2) {
```

```
        characters[k] = rightArray[j];
        consonant_count[k] = rightConsonantCount[j];
        j++;
        k++;
    }
}

void mergeSort(char characters[], int left, int right, int consonant_count[]) {
    if (left < right) {
        int middle = left + (right - left) / 2;

        mergeSort(characters, left, middle, consonant_count);
        mergeSort(characters, middle + 1, right, consonant_count);

        merge(characters, left, middle, right, consonant_count);
    }
}

int main() {
    int n;
    cin >> n;

    char characters[MAX_N];
    int consonant_count[MAX_N];

    for (int i = 0; i < n; i++) {
        cin >> characters[i];
        consonant_count[i] = isConsonant(characters[i]);
    }

    mergeSort(characters, 0, n - 1, consonant_count);

    for (int i = 0; i < n; i++) {
        cout << characters[i] << " ";
    }

    return 0;
}
```

```
#include <stdio.h>

int isConsonant(char ch) {
    if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {
        if (ch != 'a' && ch != 'e' && ch != 'i' && ch != 'o' && ch != 'u' &&
            ch != 'A' && ch != 'E' && ch != 'I' && ch != 'O' && ch != 'U') {
            return 1;
        }
    }
    return 0;
}

void merge(char characters[], int left, int middle, int right, int
consonant_count[]) {
    int n1 = middle - left + 1;
    int n2 = right - middle;

    char leftArray[n1];
    char rightArray[n2];
    int leftConsonantCount[n1];
    int rightConsonantCount[n2];

    for (int i = 0; i < n1; i++) {
        leftArray[i] = characters[left + i];
        leftConsonantCount[i] = consonant_count[left + i];
    }
    for (int j = 0; j < n2; j++) {
        rightArray[j] = characters[middle + 1 + j];
        rightConsonantCount[j] = consonant_count[middle + 1 + j];
    }

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (leftConsonantCount[i] <= rightConsonantCount[j]) {
            characters[k] = leftArray[i];
            consonant_count[k] = leftConsonantCount[i];
            i++;
        } else {
            characters[k] = rightArray[j];
            consonant_count[k] = rightConsonantCount[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        characters[k] = leftArray[i];
        consonant_count[k] = leftConsonantCount[i];
        i++;
        k++;
    }

    while (j < n2) {
        characters[k] = rightArray[j];
        consonant_count[k] = rightConsonantCount[j];
        j++;
```

```
        k++;
    }
}

void mergeSort(char characters[], int left, int right, int consonant_count[]) {
    if (left < right) {
        int middle = left + (right - left) / 2;

        mergeSort(characters, left, middle, consonant_count);
        mergeSort(characters, middle + 1, right, consonant_count);

        merge(characters, left, middle, right, consonant_count);
    }
}

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

    char characters[n];
    int consonant_count[n];

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

    mergeSort(characters, 0, n - 1, consonant_count);

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

    return 0;
}
```

Q9 Test Case Input Output

```
5
2 8 98 6 4

None
```

Weightage - 10 Input Output

```
5
-78 6 4 2 3

3
```

Weightage - 10 Input Output

```
7
4 5 9 7 1 3 6

1 3 5 7 9
```

Weightage - 15 Input Output

```
7
-9 6 -3 -1 4 8 2

-9 -3 -1
```

Weightage - 15 Input Output

```
10
7 6 3 14 8 5 12 86 71 12

3 5 7 71
```

Weightage - 25 Input Output

```
10
7 9 -63 45 1 8 3 4 -91 22

-91 -63 1 3 7 9 45
```

Weightage - 25 Sample Input Sample Output

```
5
105 236 789 415 364

105 415 789
```

Sample Input Sample Output

```
4
-9 -47 56 32

-47 -9
```

Sample Input Sample Output

```
3
2 4 6

None
```

Solution

```
#include <iostream>
using namespace std;

void merge(int arr[], int left[], int left_size, int right[], 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 mergeSortOdd(int arr[], int n) {
    if (n > 1) {
        int mid = n / 2;
        int left[mid];
        int right[n - mid];

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

        mergeSortOdd(left, mid);
        mergeSortOdd(right, n - mid);

        merge(arr, left, mid, right, n - mid);
    }
}

int main() {
    int n;
    cin >> n;

    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    int oddCount = 0;
    for (int i = 0; i < n; i++) {
        if (arr[i] % 2 != 0) {
            oddCount++;
```

```
        }
    }

    if (oddCount == 0) {
        cout << "None";
    } else {
        mergeSortOdd(arr, n);
        for (int i = 0; i < n; i++) {
            if (arr[i] % 2 != 0) {
                cout << arr[i] << " ";
            }
        }
        cout << endl;
    }

    return 0;
}
```

```c
#include <stdio.h>

void merge(int arr[], int left[], int left_size, int right[], 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 mergeSortOdd(int arr[], int n) {
    if (n > 1) {
        int mid = n / 2;
        int left[mid];
        int right[n - mid];

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

        mergeSortOdd(left, mid);
        mergeSortOdd(right, n - mid);

        merge(arr, left, mid, right, n - mid);
    }
}

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


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

    int oddCount = 0;
    for (int i = 0; i < n; i++) {
        if (arr[i] % 2 != 0) {
            oddCount++;
```

```
        }
    }

    if (oddCount == 0) {
        printf("None");
    } else {
        mergeSortOdd(arr, n);
        for (int i = 0; i < n; i++) {
            if (arr[i] % 2 != 0) {
                printf("%d ", arr[i]);
            }
        }
        printf("\n");
    }

    return 0;
}
```

Q10 Test Case Input Output

```
4
9.8 1.1 3.3 7.7

Sorted Array:
1.10 3.30 7.70 9.80
```

Weightage - 10 Input Output

```
3
-1.0 -2.0 -3.0

Sorted Array:
-3.00 -2.00 -1.00
```

Weightage - 10 Input Output

```
6
-2.5 5.0 0.0 -1.5 -2.0 -3.5

Sorted Array:
-3.50 -2.50 -2.00 -1.50 0.00 5.00
```

Weightage - 15 Input Output

```
8
5.5 4.4 3.3 2.2 1.1 0.0 -1.1 -2.2

Sorted Array:
-2.20 -1.10 0.00 1.10 2.20 3.30 4.40 5.50
```

Weightage - 15 Input Output

```
10
10.1 20.2 30.3 40.4 50.5 60.6 70.7 80.8 90.9 100.0
```

```
Sorted Array:
10.10 20.20 30.30 40.40 50.50 60.60 70.70 80.80 90.90 100.00
```

Weightage - 25 Input Output

```
15
3.14 1.1 2.71 0.5 1.618 2.0 0.25 4.0 5.5 1.25 0.75 6.0 7.5 8.0 9.0

Sorted Array:
0.25 0.50 0.75 1.10 1.25 1.62 2.00 2.71 3.14 4.00 5.50 6.00 7.50 8.00 9.00
```

Weightage - 25 Sample Input Sample Output

```
5
3.14 1.1 2.71 0.5 1.618

Sorted Array:
0.50 1.10 1.62 2.71 3.14
```

Sample Input Sample Output

```
6
-2.5 5.0 0.0 -1.5 2.0 3.5

Sorted Array:
-2.50 -1.50 0.00 2.00 3.50 5.00
```

Solution

```c
#include <stdio.h>

void merge(float arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    float L[n1], R[n2];

    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];

    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++;
    }
}

void mergeSort(float 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);

    float arr[n];
    for (int i = 0; i < n; i++) {
```

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

    mergeSort(arr, 0, n - 1);

    printf("Sorted Array:\n");
    for (int i = 0; i < n; i++) {
        printf("%.2f ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

```
#include <iostream>
#include <iomanip>

using namespace std;

void merge(float arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    float L[n1], R[n2];

    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];

    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++;
    }
}

void mergeSort(float 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;
    cin >> n;
```

```
    float arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    mergeSort(arr, 0, n - 1);

    cout << "Sorted Array:" << endl;
    for (int i = 0; i < n; i++) {
        cout << fixed << setprecision(2) << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Q11 Test Case Input Output

```
6
2.5 5.0 0.0 1.5 2.0 3.5

Sorted Array:
5.00 3.50 2.50 2.00 1.50 0.00
```

Weightage - 10 Input Output

```
3
1.0 2.0 3.0

Sorted Array:
3.00 2.00 1.00
```

Weightage - 10 Input Output

```
6
2.5 5.0 0.0 14.5 2.0 5.5

Sorted Array:
14.50 5.50 5.00 2.50 2.00 0.00
```

Weightage - 15 Input Output

```
8
5.5 4.4 3.3 2.2 1.1 0.0 1.1 2.2

Sorted Array:
5.50 4.40 3.30 2.20 2.20 1.10 1.10 0.00
```

Weightage - 15 Input Output

```
10
10.1 20.2 30.3 40.4 50.5 60.6 70.7 80.8 90.9 100.0

Sorted Array:
100.00 90.90 80.80 70.70 60.60 50.50 40.40 30.30 20.20 10.10
```

Weightage - 25 Input Output

```
15
3.14 1.1 2.71 0.5 1.618 2.0 0.25 4.0 5.5 1.25 0.75 6.0 7.5 8.0 9.0

Sorted Array:
9.00 8.00 7.50 6.00 5.50 4.00 3.14 2.71 2.00 1.62 1.25 1.10 0.75 0.50 0.25
```

Weightage - 25 Sample Input Sample Output

```
4
9.8 1.1 3.3 7.7

Sorted Array:
9.80 7.70 3.30 1.10
```

Sample Input Sample Output

```
5
3.14 1.1 2.71 0.5 1.618

Sorted Array:
3.14 2.71 1.62 1.10 0.50
```

Solution

```c
#include <stdio.h>

void merge(float arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    float L[n1], R[n2];

    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];

    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++;
    }
}

void mergeSort(float 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);

    float arr[n];
    for (int i = 0; i < n; i++) {
```

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

    mergeSort(arr, 0, n - 1);

    printf("Sorted Array:\n");
    for (int i = 0; i < n; i++) {
        printf("%.2f ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

```
#include <iostream>
#include <iomanip>

using namespace std;

void merge(float arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;

    float L[n1], R[n2];

    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];

    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++;
    }
}

void mergeSort(float 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;
    cin >> n;
```

```
    float arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    mergeSort(arr, 0, n - 1);

    cout << "Sorted Array:" << endl;
    for (int i = 0; i < n; i++) {
        cout << fixed << setprecision(2) << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Q12 Test Case Input Output

```
9
1 1 6 4 5 6 1 4 1

5 4 4 6 6 1 1 1 1
```

Weightage - 15 Input Output

```
12
6 56 15 45 74 87 96 5 4 2 3 56

2 3 4 5 6 15 45 74 87 96 56 56
```

Weightage - 15 Input Output

```
15
5 98 85 84 21 45 5 5 78 5 63 21 98 78 21

45 63 84 85 78 78 98 98 21 21 21 5 5 5 5
```

Weightage - 25 Input Output

```
25
89 6 9 8 7 55 55 78 5 5 78 5 78 89 6 9 45 65 79 78 45 65 78 89 9

7 8 79 6 6 45 45 55 55 65 65 5 5 5 9 9 9 89 89 89 78 78 78 78 78
```

Weightage - 25 Input Output

```
8
3 2 5 3 5 2 1 5

1 2 2 3 3 5 5 5
```

Weightage - 10 Input Output

```
7
3 3 8 9 1 1 1
```

```
8 9 3 3 1 1 1
```

Weightage - 10 Sample Input Sample Output

```
6
1 1 2 2 2 3

3 1 1 2 2 2
```

Sample Input Sample Output

```
5
3 2 1 3 2

1 2 2 3 3
```

Solution

```
#include <stdio.h>

const int MAX_N = 100;

void merge(int arr[], int temp[], 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 j = 0; j < n2; j++) R[j] = arr[mid + 1 + j];

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

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

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

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

        mergeSort(arr, temp, left, mid);
        mergeSort(arr, temp, mid + 1, right);
        merge(arr, temp, left, mid, right);
    }
}

void countAndSortFrequency(int arr[], int n) {
    int freqArr[MAX_N][2] = {0};
    int values[MAX_N];
    int frequencies[MAX_N];

    int index = 0;
    for (int i = 0; i < n; i++) {
        int isAlreadyCounted = 0;
        for (int j = 0; j < index; j++) {
            if (values[j] == arr[i]) {
                freqArr[j][1]++;
                isAlreadyCounted = 1;
                break;
            }
        }
        if (!isAlreadyCounted) {
            values[index] = arr[i];
            freqArr[index][0] = arr[i];
            freqArr[index][1] = 1;
```

```
            index++;
        }
    }

    for (int i = 0; i < index; i++) {
        for (int j = i + 1; j < index; j++) {
            if (freqArr[i][1] > freqArr[j][1] || (freqArr[i][1] == freqArr[j][1]
&& freqArr[i][0] > freqArr[j][0])) {
                int temp = freqArr[i][0];
                freqArr[i][0] = freqArr[j][0];
                freqArr[j][0] = temp;

                temp = freqArr[i][1];
                freqArr[i][1] = freqArr[j][1];
                freqArr[j][1] = temp;
            }
        }
    }

    for (int i = 0; i < index; i++) {
        for (int j = 0; j < freqArr[i][1]; j++) {
            printf("%d ", freqArr[i][0]);
        }
    }
    printf("\n");
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[MAX_N];

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

    countAndSortFrequency(arr, n);

    return 0;
}
```

```
#include <iostream>
using namespace std;

const int MAX_N = 100;

void merge(int arr[], int temp[], 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 j = 0; j < n2; j++) R[j] = arr[mid + 1 + j];

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

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

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

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

        mergeSort(arr, temp, left, mid);
        mergeSort(arr, temp, mid + 1, right);
        merge(arr, temp, left, mid, right);
    }
}

void countAndSortFrequency(int arr[], int n) {
    int freqArr[MAX_N][2] = {0};
    int values[MAX_N];
    int frequencies[MAX_N];

    int index = 0;
    for (int i = 0; i < n; i++) {
        int isAlreadyCounted = 0;
        for (int j = 0; j < index; j++) {
            if (values[j] == arr[i]) {
                freqArr[j][1]++;
                isAlreadyCounted = 1;
                break;
            }
        }
        if (!isAlreadyCounted) {
            values[index] = arr[i];
            freqArr[index][0] = arr[i];
```

```
            freqArr[index][1] = 1;
            index++;
        }
    }

    for (int i = 0; i < index; i++) {
        for (int j = i + 1; j < index; j++) {
            if (freqArr[i][1] > freqArr[j][1] || (freqArr[i][1] == freqArr[j][1]
&& freqArr[i][0] > freqArr[j][0])) {
                swap(freqArr[i][0], freqArr[j][0]);
                swap(freqArr[i][1], freqArr[j][1]);
            }
        }
    }

    for (int i = 0; i < index; i++) {
        for (int j = 0; j < freqArr[i][1]; j++) {
            cout << freqArr[i][0] << " ";
        }
    }
    cout << endl;
}

int main() {
    int n;
    cin >> n;
    int arr[MAX_N];

    for (int i = 0; i < n; i++) cin >> arr[i];

    countAndSortFrequency(arr, n);

    return 0;
}
```