1-Number of Zeros in a Given Array

Started on	Monday, 22 September 2025, 7:33 PM
State	Finished
Completed on	Monday, 22 September 2025, 7:36 PM
Time taken	2 mins 32 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1 Correct Mark 1.00 out of 1.00 Flag question

Problem Statement

Given an array of 1s and 0s this has all 1s first followed by all 0s. Aim is to find the number of 0s. Write a program using D
of zeroes in the given array.

Input Format

First Line Contains Integer m – Size of array

Next m lines Contains m numbers – Elements of an array

Output Format

First Line Contains Integer – Number of zeroes present in the given array.

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int firstZeroIndex(int arr[], int low, int high) {
    if (high >= low) {
        int mid = (low + high) / 2;
        if ((mid == 0 || arr[mid - 1] == 1) && arr[mid] == 0)
            return mid;
        if (arr[mid] == 1)
            return firstZeroIndex(arr, mid + 1, high);
        return firstZeroIndex(arr, low, mid - 1);
    }
    return -1;
}
int countZeroes(int arr[], int n) {
    int first = firstZeroIndex(arr, 0, n - 1);

    if (first == -1)
        return 0;
    return (n - first);
}
int main() {
    int m;
    scanf("%d", &m);

    int arr[m];
    for (int i = 0; i < m; i++) {
        scanf("%d", &arr[i]);
    }
    int result = countZeroes(arr, m);
    printf("%d\n", result);
    return 0;
}

```

	Input	Expected	Got	
✓	5	2	2	✓
	1 1			

2-Majority Element

Started on	Monday, 22 September 2025, 7:36 PM
State	Finished
Completed on	Monday, 22 September 2025, 7:37 PM
Time taken	1 min 14 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1 | Correct Mark 1.00 out of 1.00 Flag question

Given an array `nums` of size `n`, return *the majority element*.

The majority element is the element that appears more than `⌊n / 2⌋` times. You may assume that the majority element always ex

Example 1:

```
Input: nums = [3,2,3]
Output: 3
```

Example 2:

```
Input: nums = [2,2,1,1,1,2,2]
Output: 2
```

Constraints:

- `n == nums.length`
- `1 <= n <=` $5 * 10^4$
- -2^{31} `<= nums[i] <=` $2^{31} - 1$

For example:

Input	Result
3 3 2 3	3
7 2 2 1 1 1 2 2	2

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int majorityElement(int* nums, int n) {
    int count = 0, candidate = 0;
    for (int i = 0; i < n; i++) {
        if (count == 0) {
            candidate = nums[i];
        }
        count += (nums[i] == candidate) ? 1 : -1;
    }
    return candidate;
}

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

	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

3-Finding Floor Value

Started on	Monday, 22 September 2025, 7:38 PM
State	Finished
Completed on	Monday, 22 September 2025, 7:39 PM
Time taken	1 min 13 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1 | Correct Mark 1.00 out of 1.00 Flag question

Problem Statement:

Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write divide ar

Input Format

First Line Contains Integer n – Size of array

Next n lines Contains n numbers – Elements of an array

Last Line Contains Integer x – Value for x

Output Format

First Line Contains Integer – Floor value for x

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int floorSearch(int arr[], int low, int high, int x) {
    if (low > high) return -1;
    if (x >= arr[high]) return arr[high];

    int mid = (low + high) / 2;

    if (arr[mid] == x) return arr[mid];

    if (mid > 0 && arr[mid - 1] <= x && x < arr[mid])
        return arr[mid - 1];

    if (x < arr[mid])
        return floorSearch(arr, low, mid - 1, x);

    return floorSearch(arr, mid + 1, high, x);
}

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

    int result = floorSearch(arr, 0, n - 1, x);
    printf("%d\n", result);

    return 0;
}
```

	Input	Expected	Got	

Dashboard My courses

CS23331-DAA-2024-CSE / 4-Two Elements sum to x

4-Two Elements sum to x

Started on	Monday, 22 September 2025, 7:40 PM
State	Finished
Completed on	Monday, 22 September 2025, 7:40 PM
Time taken	37 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1 Correct Mark 1.00 out of 1.00 Flag question

Problem Statement:

Given a sorted array of integers say arr[] and a number x. Write a recursive program using divide and conquer strateg
array whose sum = x. If there exist such two elements then return the numbers, otherwise print as "No".

Note: Write a Divide and Conquer Solution

Input Format

First Line Contains Integer n – Size of array

Next n lines Contains n numbers – Elements of an array

Last Line Contains Integer x – Sum Value

Output Format

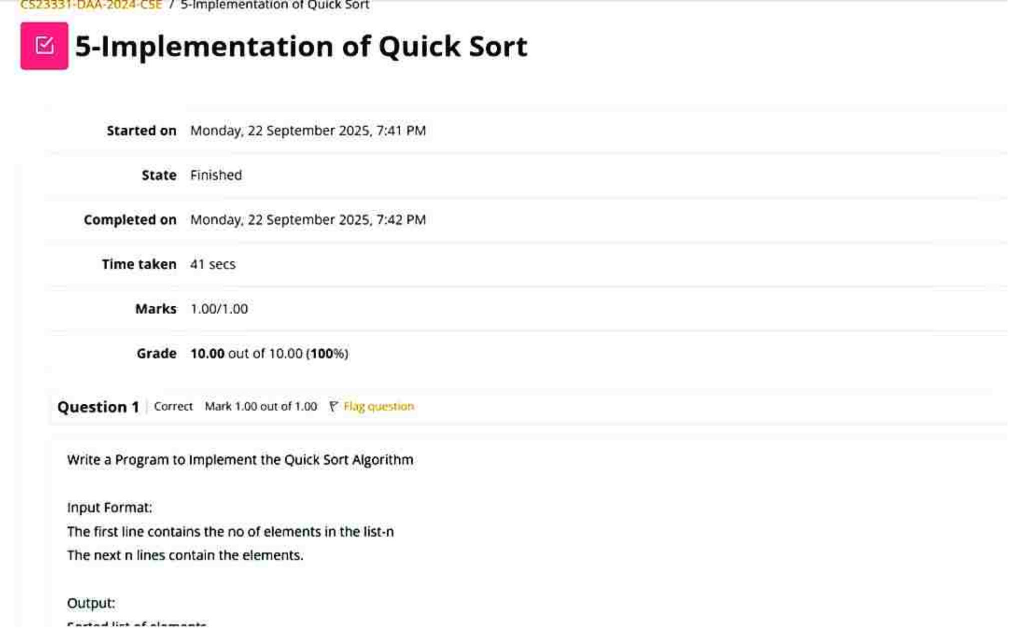
First Line Contains Integer – Element1

Second Line Contains Integer – Element2 (Element 1 and Elements 2 together sums to value "x")

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int findPair(int arr[], int low, int high, int x, int *a, int *b) {
    if (low >= high) return 0;
    int sum = arr[low] + arr[high];

    if (sum == x) {
        *a = arr[low];
        *b = arr[high];
        return 1;
    }
    else if (sum > x) {
        return findPair(arr, low, high - 1, x, a, b);
    } else {
        return findPair(arr, low + 1, high, x, a, b);
    }
}

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

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

    int x;
    scanf("%d", &x);

    int a, b;
    if (findPair(arr, 0, n - 1, x, &a, &b)) {
        printf("%d\n%d\n", a, b);
    } else {
        printf("No\n");
    }

    return 0;
}
```

	Input	Expected	Got	



5-Implementation of Quick Sort

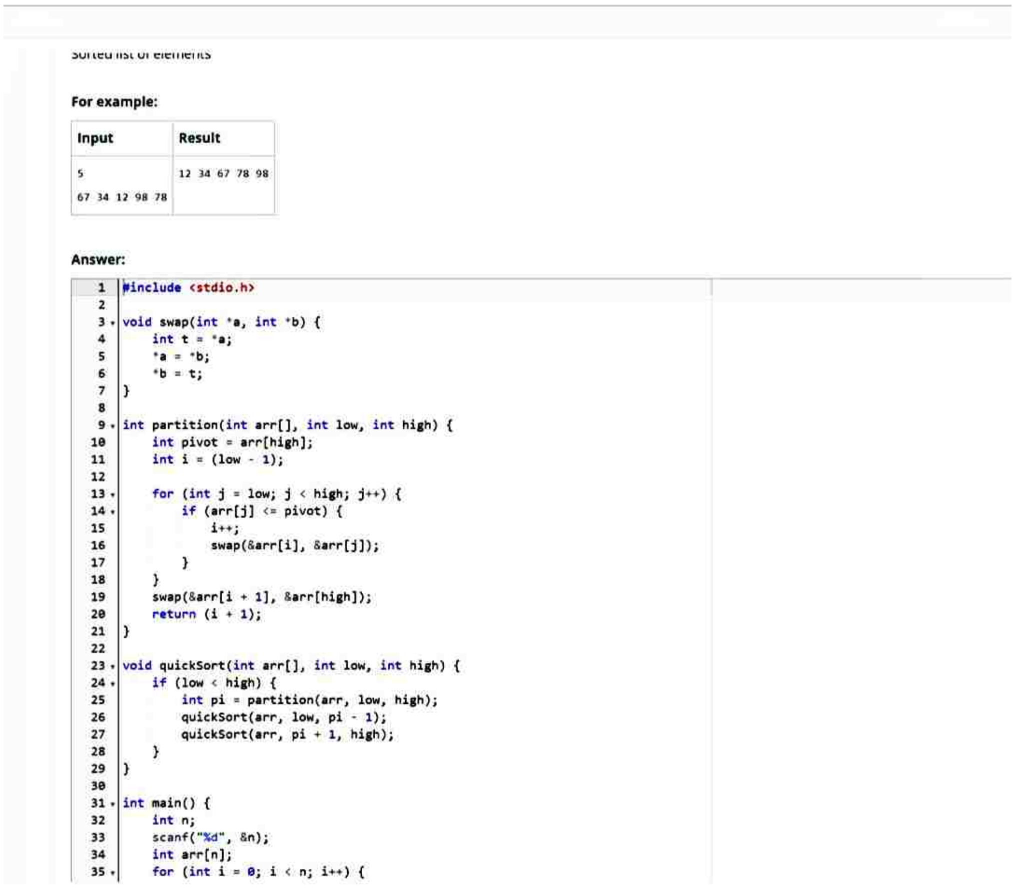
Started on	Monday, 22 September 2025, 7:41 PM
State	Finished
Completed on	Monday, 22 September 2025, 7:42 PM
Time taken	41 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1 Correct Mark 1.00 out of 1.00 Flag question

Write a Program to Implement the Quick Sort Algorithm

Input Format:
The first line contains the no of elements in the list-n
The next n lines contain the elements.

Output:



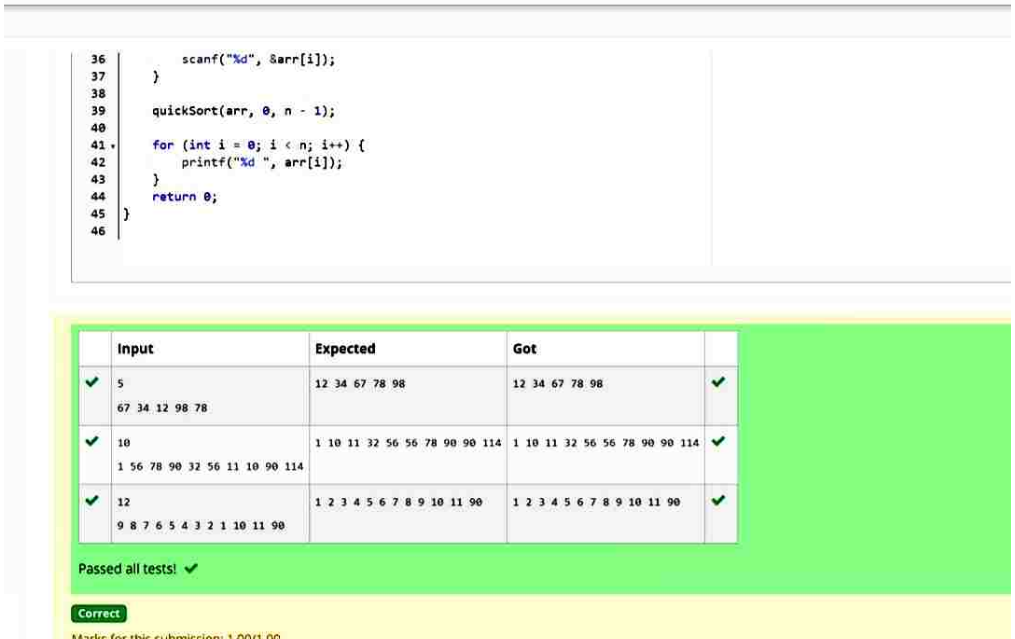
Sorted list of elements

For example:

Input	Result
5 67 34 12 98 78	12 34 67 78 98

Answer:

```c
#include <stdio.h>

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

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



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

    quickSort(arr, 0, n - 1);

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

```

	Input	Expected	Got	
✓	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98	✓
✓	10 1 56 78 90 32 56 11 10 90 114	1 10 11 32 56 56 78 90 90 114	1 10 11 32 56 56 78 90 90 114	✓
✓	12 9 8 7 6 5 4 3 2 1 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.