

CS23331-DAA-2024-CSE / 1-Number of Zeros in a Given Array

1-Number of Zeros in a Given Array

Started on	Friday, 19 September 2025, 1:30 PM
State	Finished
Completed on	Friday, 19 September 2025, 1:46 PM
Time taken	16 mins 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](#)

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 Divide and Conquer to Count the number 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 %)

```
1 #include <stdio.h>
2 int findFirstZero(int arr[], int low, int high) {
3     int firstZeroIndex = -1;
4     while (low <= high) {
5         int mid = low + (high - low) / 2;
6
7         if (arr[mid] == 0) {
8             firstZeroIndex = mid;
9             high = mid - 1;
10        } else {
11            low = mid + 1;
12        }
13    }
14    return firstZeroIndex;
15 }
16 int countZeroes(int arr[], int size) {
17     int firstZeroIndex = findFirstZero(arr, 0, size - 1);
18     if (firstZeroIndex == -1) {
19         return 0;
20     }
21     return size - firstZeroIndex;
22 }
23
24 int main() {
25     int m;
26     scanf("%d", &m);
27     int arr[m];
28     for (int i = 0; i < m; i++) {
29         scanf("%d", &arr[i]);
30     }
31     int zeroCount = countZeroes(arr, m);
32     printf("%d\n", zeroCount);
33     return 0;
34 }
35
```

	Input	Expected	Got	
✓	5 1 1 1 0 0	2	2	✓
✓	10 1 1 1 1 1 1 1 1 1 1	0	0	✓

2-Majority Element

Started on	Friday, 19 September 2025, 1:49 PM
State	Finished
Completed on	Friday, 19 September 2025, 2:29 PM
Time taken	39 mins 8 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 $\lfloor n / 2 \rfloor$ times. You may assume that the majority element always exists in the array.

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`
- $1 \leq n \leq 5 \cdot 10^4$
- $-2^{31} \leq \text{nums}[i] \leq 2^{31} - 1$

For example:

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

Answer: (penalty regime: 0 %)

```

1 #include <stdio.h>
2
3 int main() {
4     int n;
5     scanf("%d", &n);
6     int nums[n];
7     for (int i = 0; i < n; i++) {
8         scanf("%d", &nums[i]);
9     }
10
11     int candidate = 0;
12     int count = 0;
13
14     for (int i = 0; i < n; i++) {
15         if (count == 0) {
16             candidate = nums[i];
17             count = 1;
18         } else if (nums[i] == candidate) {
19             count++;
20         } else {
21             count--;
22         }
23     }
24
25     printf("%d\n", candidate);
26
27     return 0;
28 }

```

	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

[Finish review](#)

[Back to Course](#)

3-Finding Floor Value

Started on	Sunday, 12 October 2025, 9:13 AM
State	Finished
Completed on	Sunday, 12 October 2025, 9:22 AM
Time taken	9 mins 23 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 and conquer algorithm to find floor of x.

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 %)

```

1 #include<stdio.h>
2 int findFloor(int arr[],int n,int x){
3     int low=0;
4     int high=n-1;
5     int floor_val=-1;
6
7     if(x<arr[0])
8         return -1;
9     while(low<=high){
10         int mid=low+(high-low)/2;
11         if(arr[mid]==x)
12             return arr[mid];
13         if(arr[mid]<x){
14             floor_val=arr[mid];
15             low=mid+1;
16         }
17         else{
18             high=mid-1;
19         }
20     }
21     return floor_val;
22 }
23 int main(){
24     int n;
25     scanf("%d",&n);
26     int arr[n];
27     for(int i=0;i<n;i++){
28         scanf("%d",&arr[i]);
29     }
30     int x;
31     scanf("%d",&x);
32     int result=findFloor(arr,n,x);
33     if(result== -1)
34         printf("Floor does not exist\n");
35     else
36         printf("%d\n",result);
37     return 0;
38 }
39

```

	Input	Expected	Got	
✓	6 1 2 8 10 12 19 5	2	2	✓
✓	5 10 22 85 108	85	85	✓

	129 100			
✓	7 3 5 7 9 11 13 15 10	9	9	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course



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



4-Two Elements sum to x

✔ Done

Opened: Sunday, 14 April 2024, 12:17 PM

Re-attempt quiz

Grading method: Highest grade

Summary of your previous attempts

Attempt	State	Marks / 1.00	Grade / 10.00	Review
1	Finished Submitted Sunday, 12 October 2025, 9:32 AM	1.00	10.00	Review

Highest grade: 10.00 / 10.00.

Back to Course



CS23331-DAA-2024-CSE / 5-Implementation of Quick Sort



5-Implementation of Quick Sort

✔ Done

Opened: Thursday, 30 May 2024, 9:49 AM

Re-attempt quiz

Attempts allowed: 2

Grading method: Highest grade

Summary of your previous attempts

Attempt	State	Marks / 1.00	Grade / 10.00	Review
1	Finished Submitted Sunday, 12 October 2025, 9:41 AM	1.00	10.00	Review

Highest grade: 10.00 / 10.00.

Back to Course