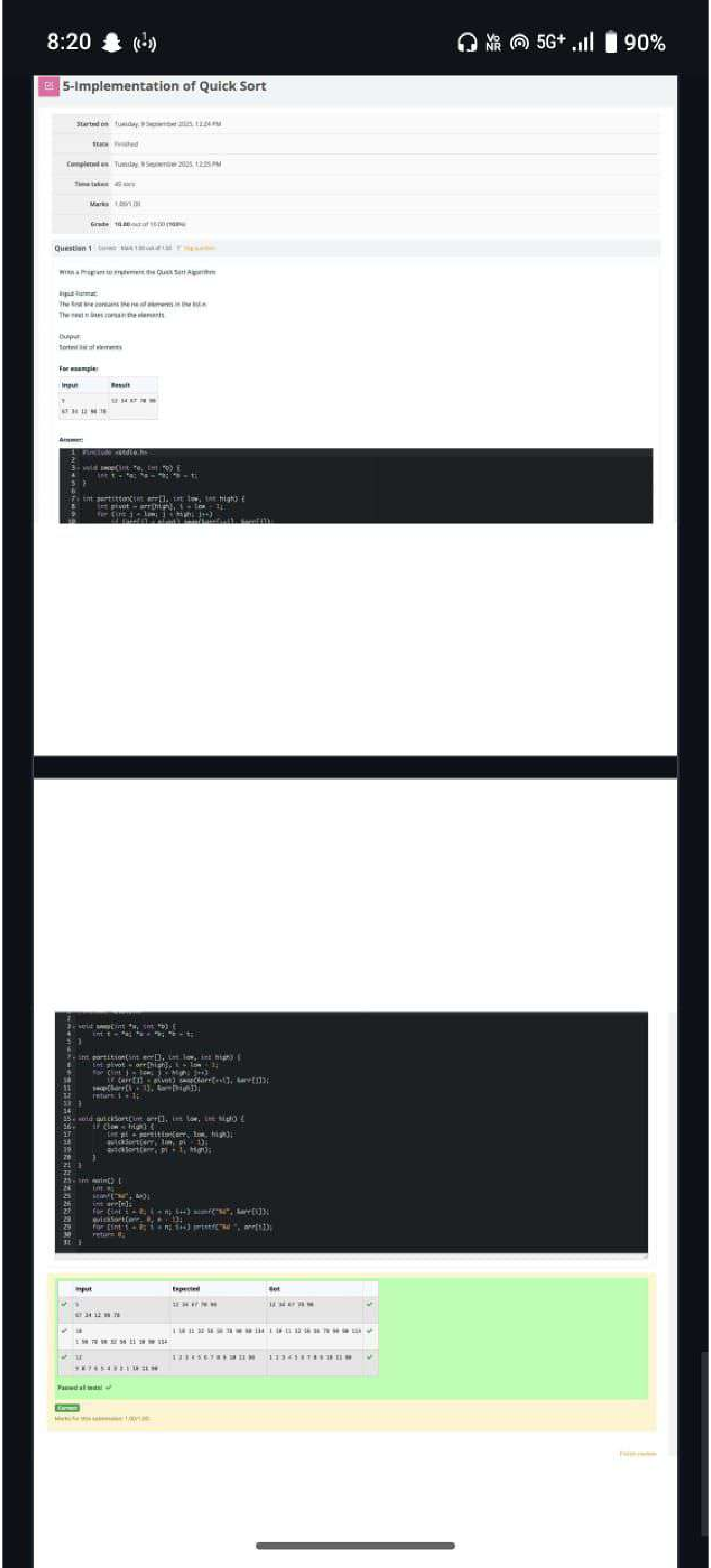


5-Implementation of Quick Sort

Started on	Tuesday, 9 September 2025, 12:24 PM
State	Finished
Completed on	Tuesday, 9 September 2025, 12:25 PM
Time taken	45 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], i = low - 1;
    for (int j = low; j < high; j++)
```

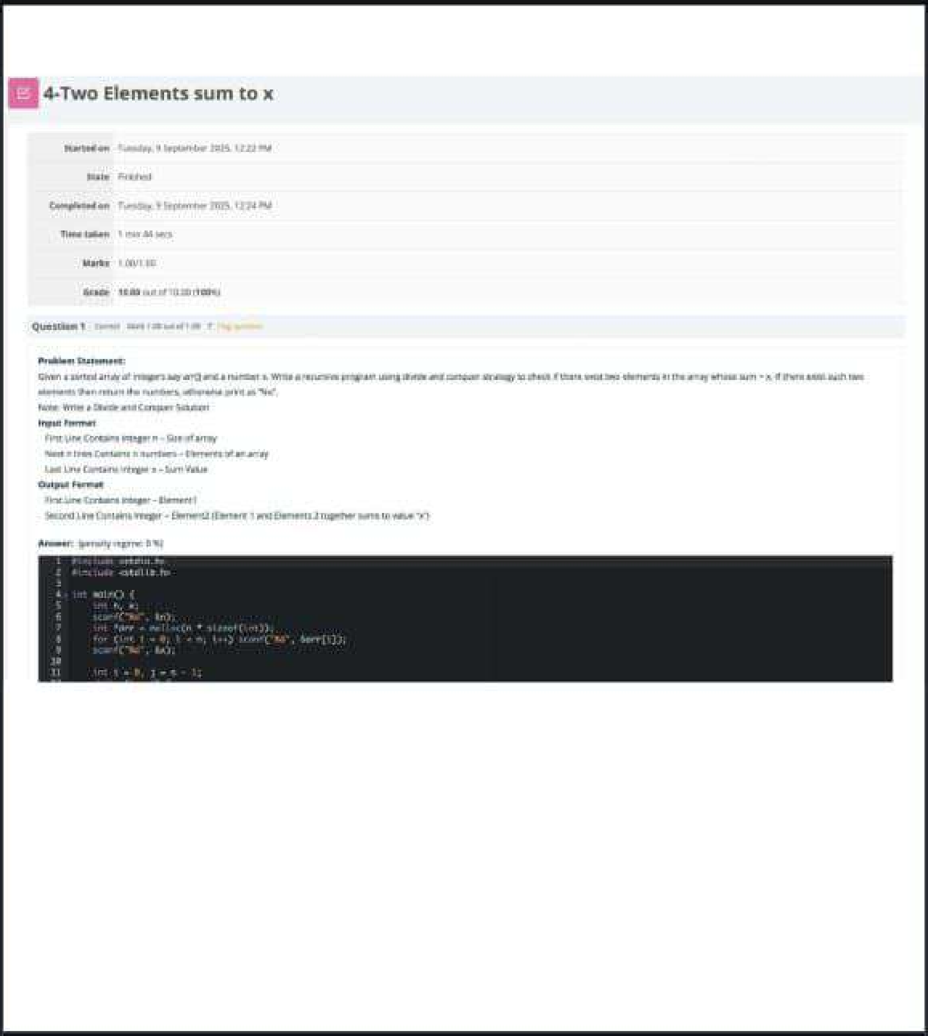
```c

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], i = low - 1;
    for (int j = low; j < high; j++)
        if (arr[j] < pivot) 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++) 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 98 32 56 11 10 90 114	1 10 11 32 56 56 78 90 98 114	1 10 11 32 56 56 78 90 98 114	✓
✓	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.

Finish review



4-Two Elements sum to x

Started on	Tuesday, 9 September 2025, 12:22 PM
State	Finished
Completed on	Tuesday, 9 September 2025, 12:24 PM
Time taken	1 min 44 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 strategy to check if there exist two elements in the 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

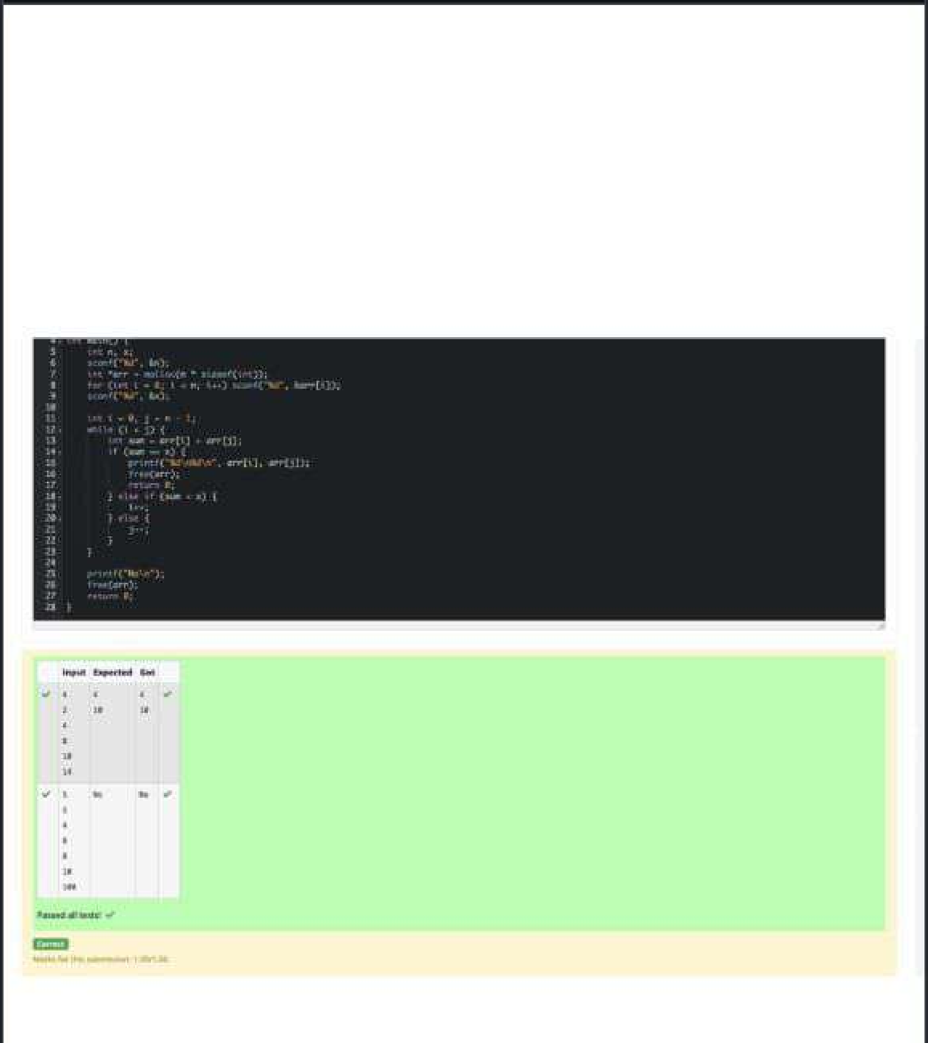
Last Line Contains integer x – Sum Value

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 %)



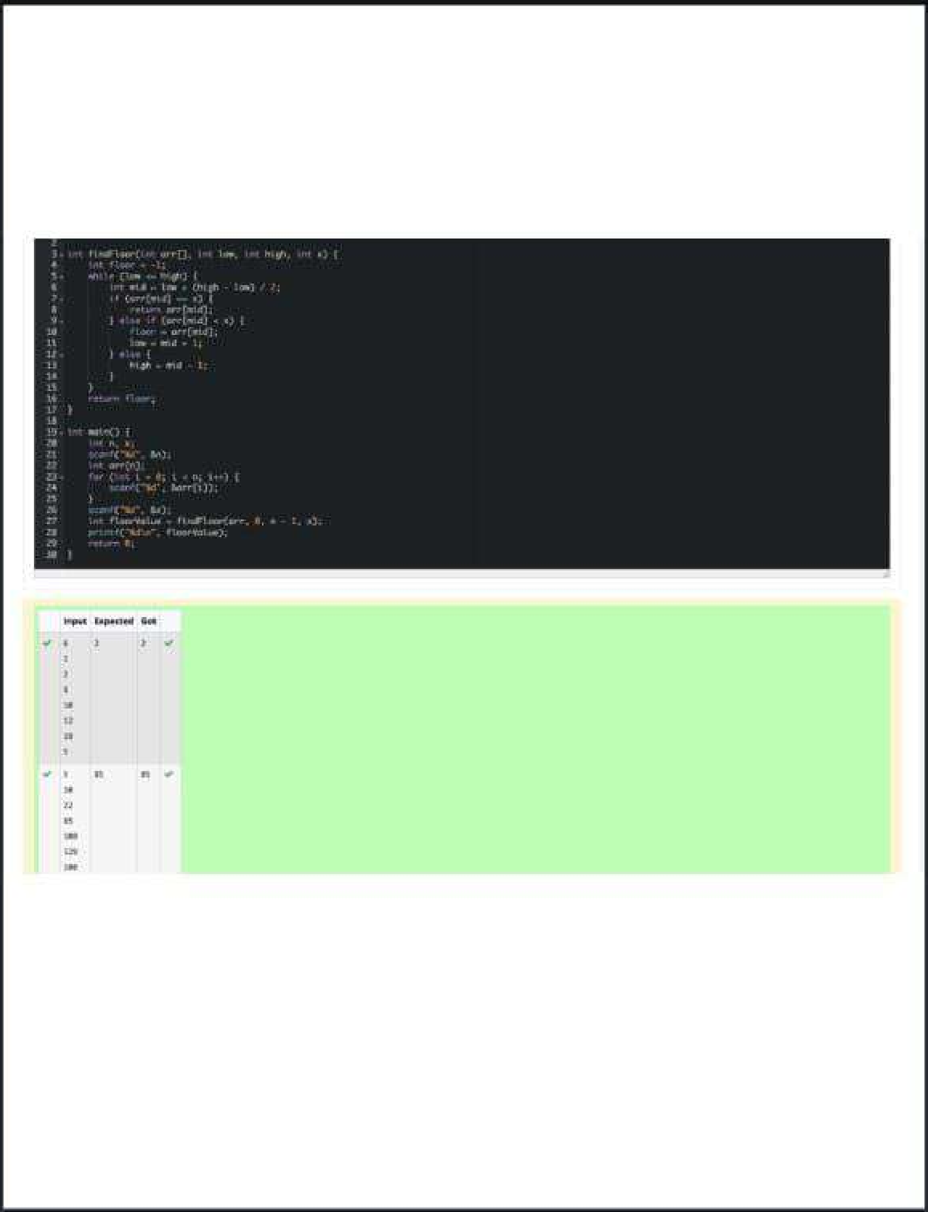
```c
#include <stdio.h>
#include <stdlib.h>

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

    int i = 0, j = n - 1;
    while (i < j) {
        int sum = arr[i] + arr[j];
        if (sum == x) {
            printf("%d\n%d\n", arr[i], arr[j]);
            free(arr);
            return 0;
        } else if (sum < x) {
            i++;
        } else {
            j--;
        }
    }

    printf("No\n");
    free(arr);
    return 0;
}
```

	Input	Expected	Got	
✓	4 2 4 8 10 14	4 10	4 10	✓
✓	5 2 4 6 8 10 100	No	No	✓

Passed all tests! ✓

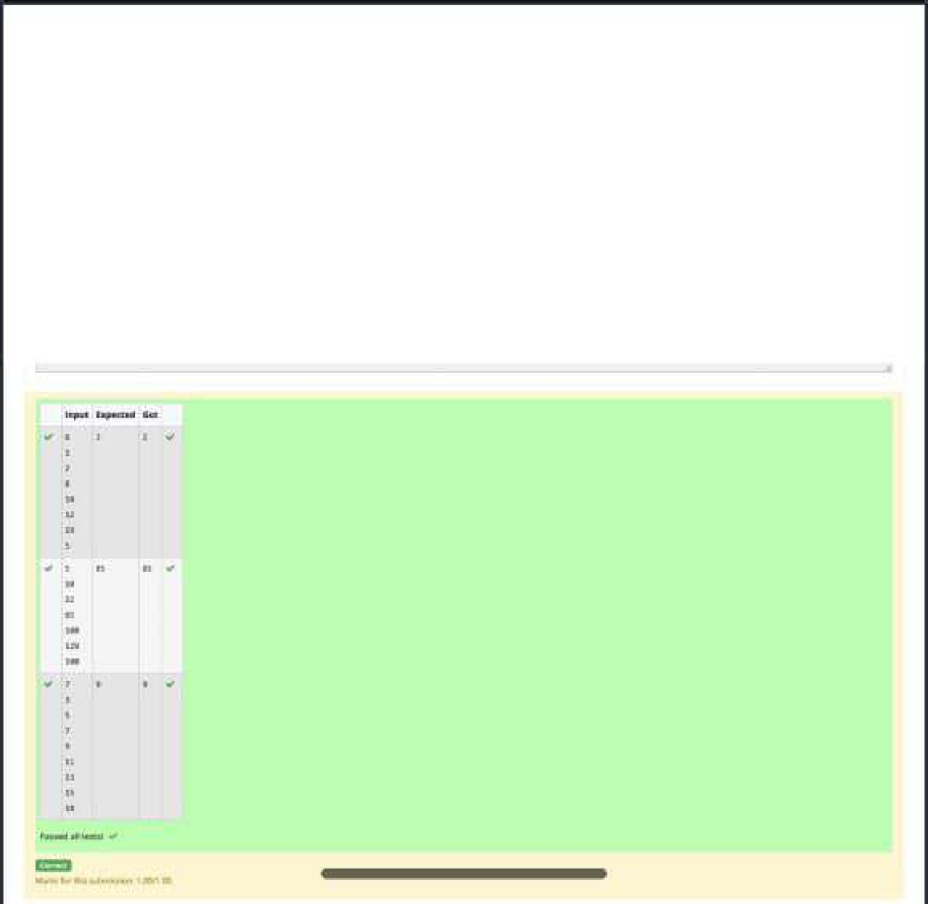
Correct

Marks for this submission: 1.00/1.00.



```c
int findFloor(int arr[], int low, int high, int x) {
    int floor = -1;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == x) {
            return arr[mid];
        } else if (arr[mid] < x) {
            floor = arr[mid];
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return floor;
}

int main() {
    int n, x;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    scanf("%d", &x);
    int floorValue = findFloor(arr, 0, n - 1, x);
    printf("%d\n", floorValue);
    return 0;
}
```

	Input	Expected	Got	
✓	6 1 2 8 10 12 19 5	2	2	✓
✓	5 10 22 85 100 120 100	85	85	✓

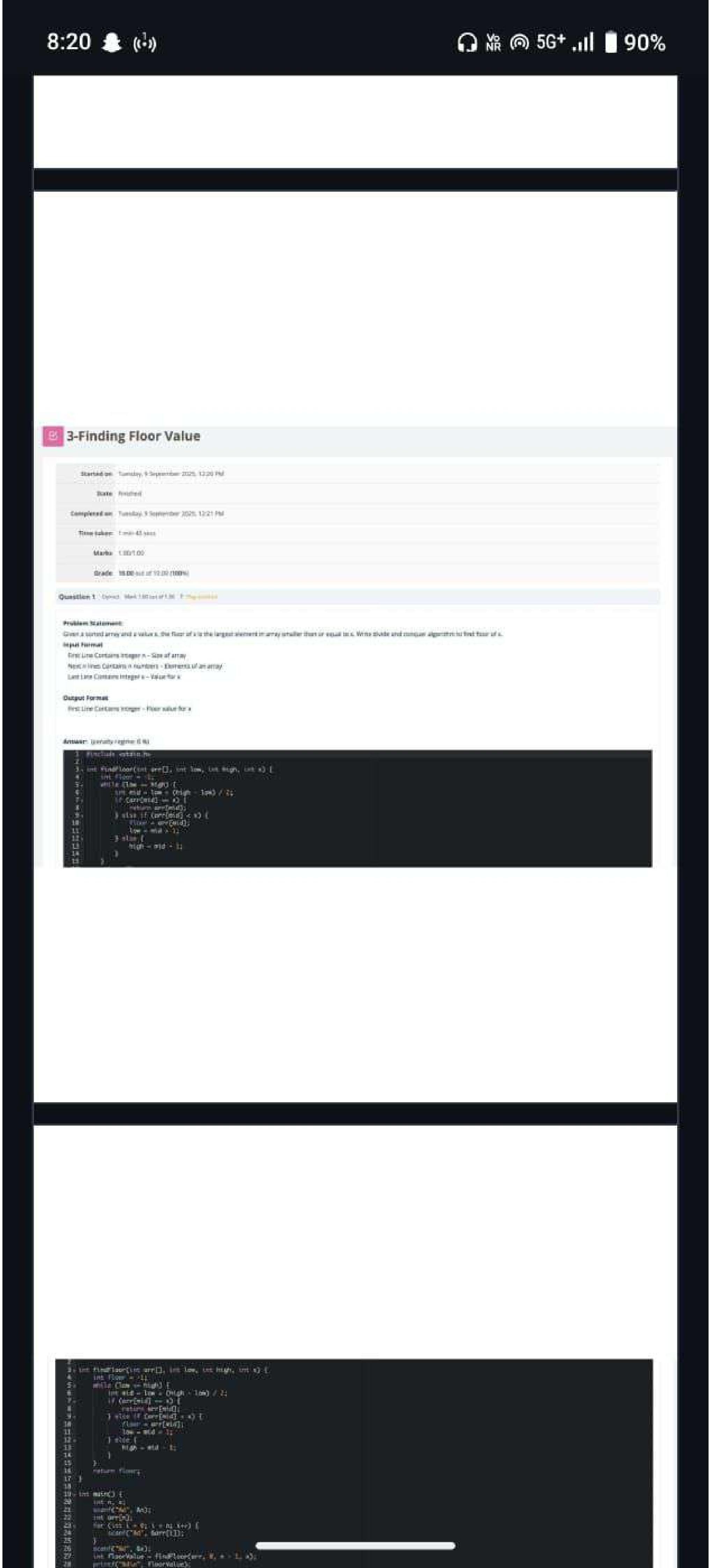


	Input	Expected	Got	
✓	6 1 2 8 10 12 19 5	2	2	✓
✓	5 10 22 85 100 120 100	85	85	✓
✓	7 3 5 7 9 11 13 15 10	9	9	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



8:20 5G+ 90%

3-Finding Floor Value

Started on	Tuesday, 9 September 2025, 12:20 PM
State	Finished
Completed on	Tuesday, 9 September 2025, 12:21 PM
Time taken	1 min 43 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

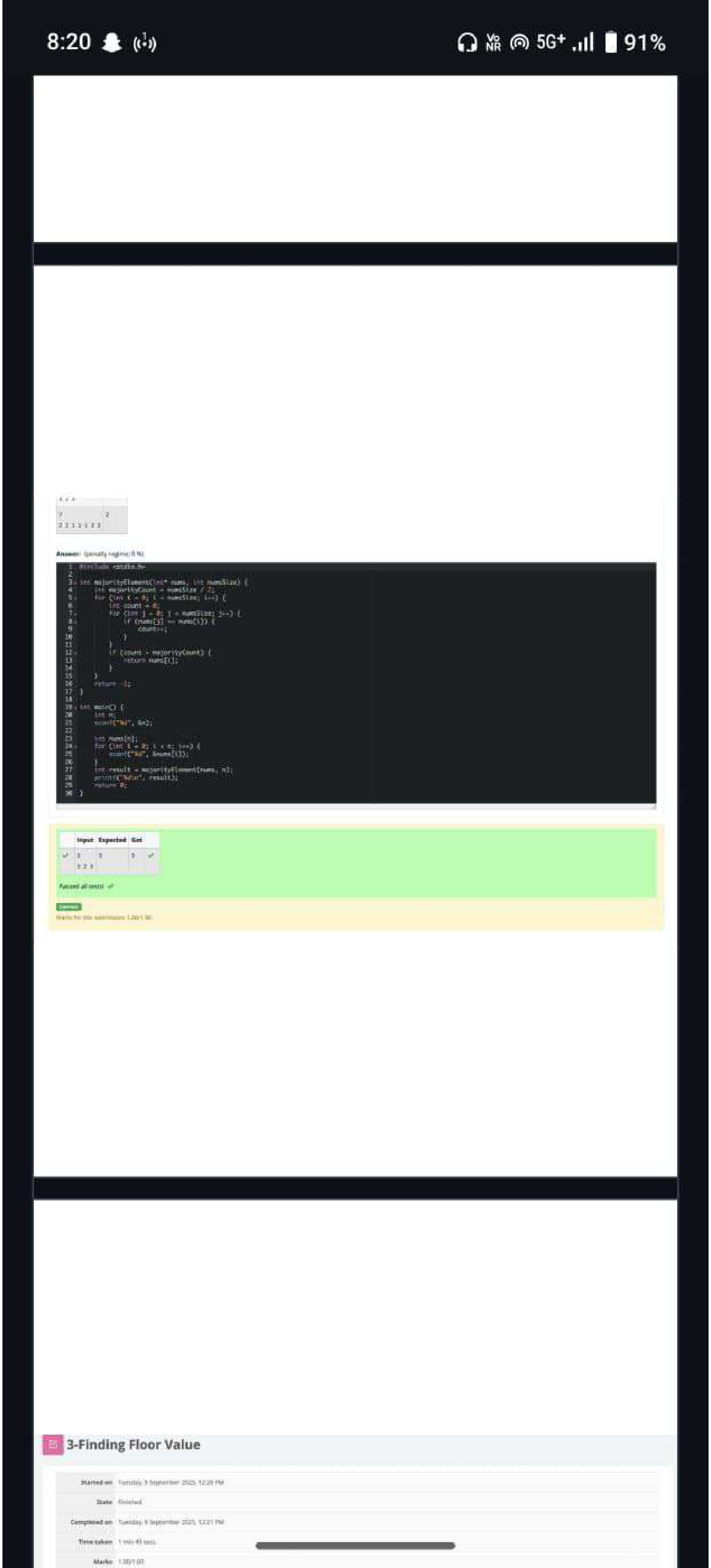
Problem Statement:
Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write divide and conquer algorithm to find floor of x.
Input Format
First Line Contains Integer n – Size of array
Next n lines Contains n numbers – Elements of an array
Last Line Contains Integer x – Value for x

Output Format
First Line Contains Integer – Floor value for x

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int findFloor(int arr[], int low, int high, int x) {
    int floor = -1;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == x) {
            return arr[mid];
        } else if (arr[mid] < x) {
            floor = arr[mid];
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
```

```c
int findFloor(int arr[], int low, int high, int x) {
    int floor = -1;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == x) {
            return arr[mid];
        } else if (arr[mid] < x) {
            floor = arr[mid];
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return floor;
}

int main() {
    int n, x;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    scanf("%d", &x);
    int floorValue = findFloor(arr, 0, n - 1, x);
    printf("%d\n", floorValue);
```



8:20 91%

```
4 4 4
```

7 2 2 1 1 1 2 2	2

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int majorityElement(int* nums, int numsSize) {
    int majorityCount = numsSize / 2;
    for (int i = 0; i < numsSize; i++) {
        int count = 0;
        for (int j = 0; j < numsSize; j++) {
            if (nums[j] == nums[i]) {
                count++;
            }
        }
        if (count > majorityCount) {
            return nums[i];
        }
    }
    return -1;
}

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

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

	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

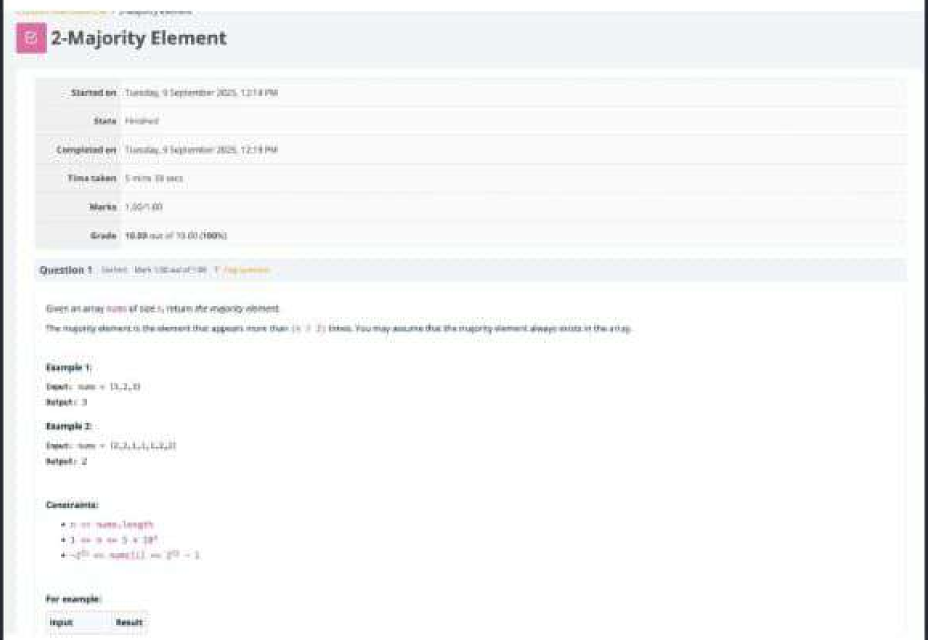
Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

3-Finding Floor Value

Started on	Tuesday, 9 September 2025, 12:20 PM
State	Finished
Completed on	Tuesday, 9 September 2025, 12:21 PM
Time taken	1 min 48 secs
Marks	1.00/1.00



2-Majority Element

Started on	Tuesday, 9 September 2025, 12:14 PM
State	Finished
Completed on	Tuesday, 9 September 2025, 12:19 PM
Time taken	5 mins 39 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 exists in the array.

Example 1:

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

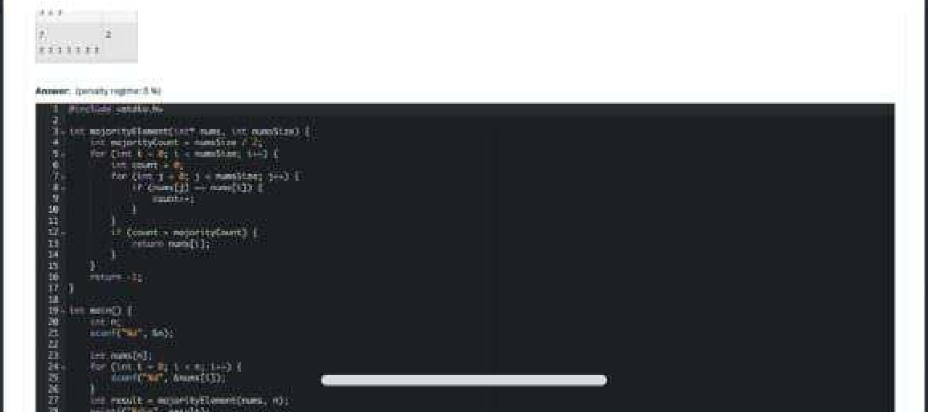
Example 2:

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

Constraints:

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

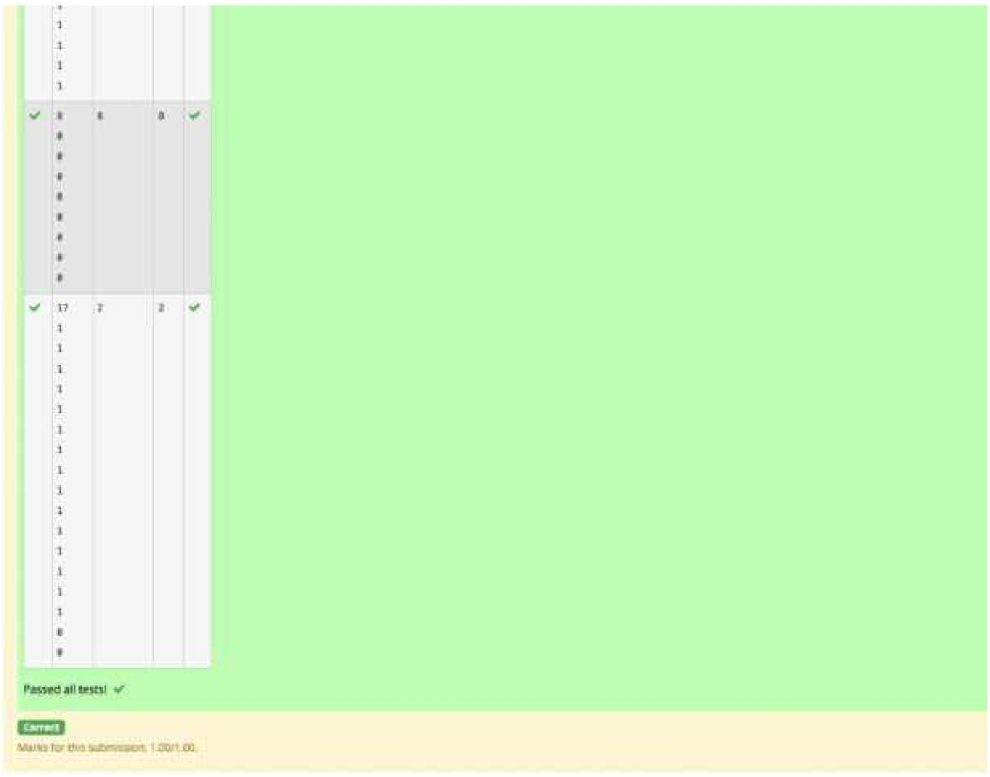
For example:



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

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int majorityElement(int* nums, int numsSize) {
    int majorityCount = numsSize / 2;
    for (int i = 0; i < numsSize; i++) {
        int count = 0;
        for (int j = 0; j < numsSize; j++) {
            if (nums[j] == nums[i]) {
                count++;
            }
        }
        if (count > majorityCount) {
            return nums[i];
        }
    }
    return -1;
}

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

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

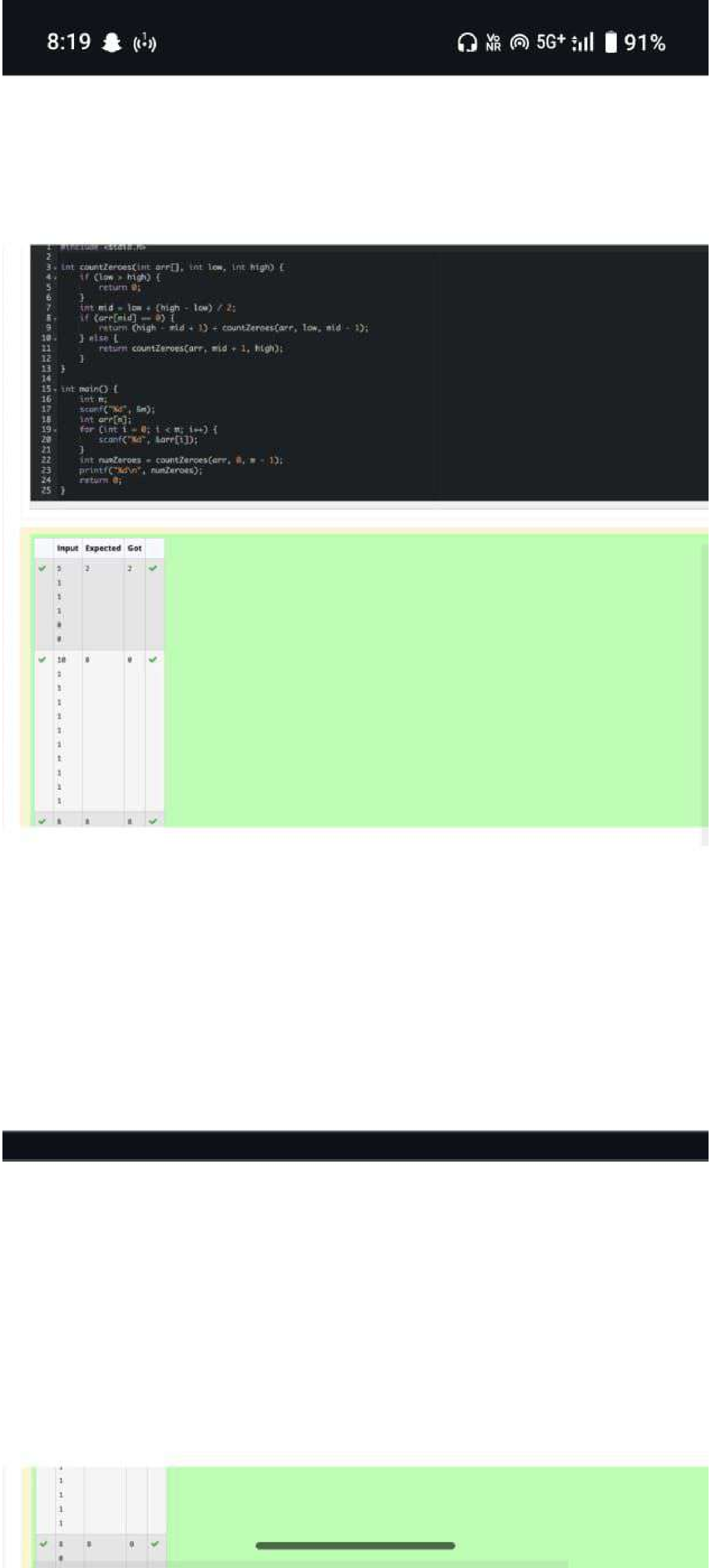


	1			
	1			
	1			
	1			
✓	8	8	8	✓
	0			
	0			
	0			
	0			
	0			
	0			
	0			
	0			
✓	17	2	2	✓
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	1			
	0			
	0			

Passed all tests! ✓

Correct

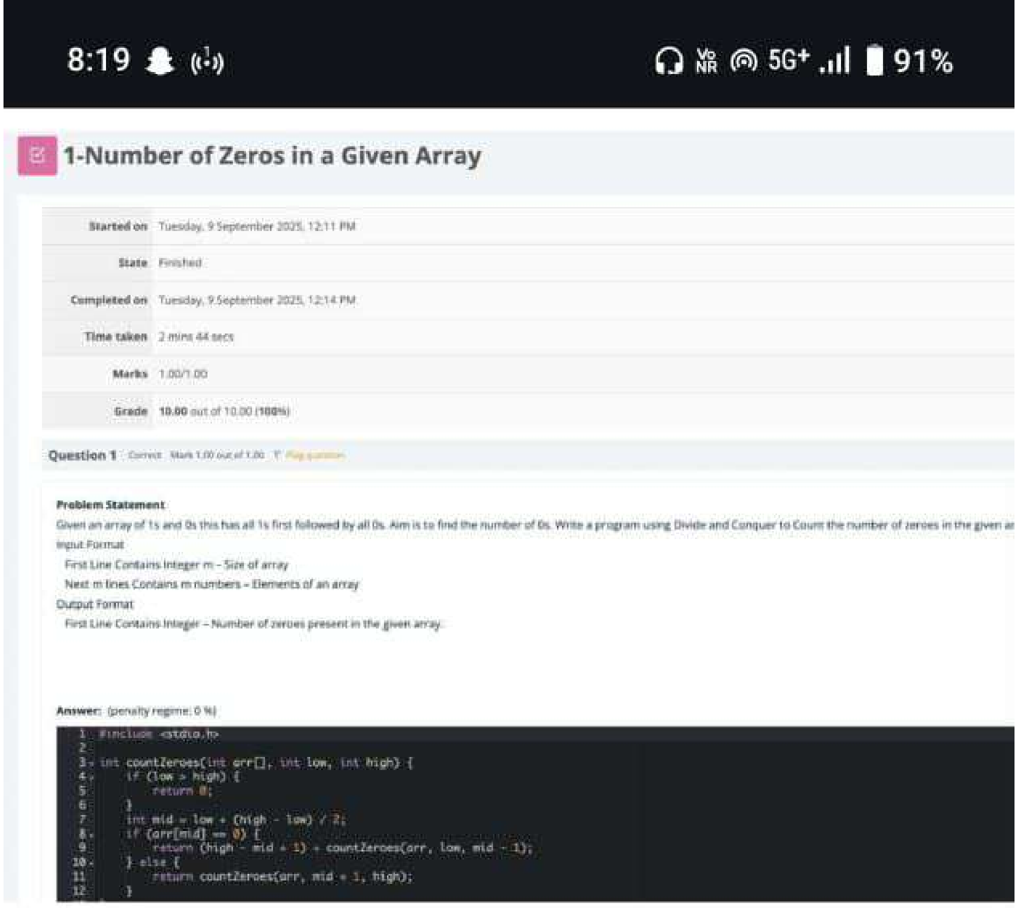
Marks for this submission: 1.00/1.00.



```c
#include <stdio.h>

int countZeroes(int arr[], int low, int high) {
    if (low > high) {
        return 0;
    }
    int mid = low + (high - low) / 2;
    if (arr[mid] == 0) {
        return (high - mid + 1) + countZeroes(arr, low, mid - 1);
    } else {
        return countZeroes(arr, mid + 1, high);
    }
}

int main() {
    int m;
    scanf("%d", &m);
    int arr[m];
    for (int i = 0; i < m; i++) {
        scanf("%d", &arr[i]);
    }
    int numZeroes = countZeroes(arr, 0, m - 1);
    printf("%d\n", numZeroes);
    return 0;
}
```

	Input	Expected	Got	
✓	5 1 1 1 0 0	2	2	✓
✓	10 1 1 1 1 1 1 1 1 1 1	0	0	✓
✓	8	8	8	✓

	1 1 1 1 1			
✓	8 0	8	8	✓



8:19 5G+ 91%

1-Number of Zeros in a Given Array

Started on	Tuesday, 9 September 2025, 12:11 PM
State	Finished
Completed on	Tuesday, 9 September 2025, 12:14 PM
Time taken	2 mins 44 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 ar

Input Format

First Line Contains Integer m – Size of array

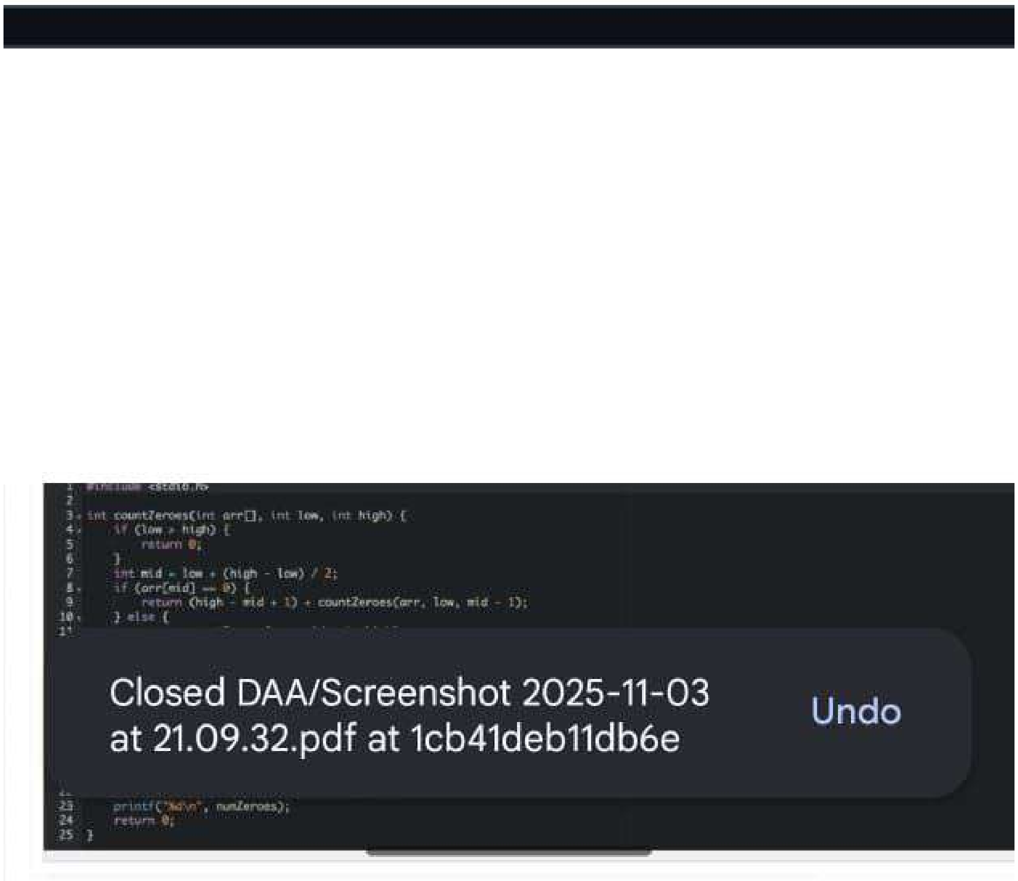
Next m lines Contains m numbers – Elements of an array

Output Format

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

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int countZeroes(int arr[], int low, int high) {
    if (low > high) {
        return 0;
    }
    int mid = low + (high - low) / 2;
    if (arr[mid] == 0) {
        return (high - mid + 1) + countZeroes(arr, low, mid - 1);
    } else {
        return countZeroes(arr, mid + 1, high);
    }
```



```c
#include <stdio.h>

int countZeroes(int arr[], int low, int high) {
    if (low > high) {
        return 0;
    }
    int mid = low + (high - low) / 2;
    if (arr[mid] == 0) {
        return (high - mid + 1) + countZeroes(arr, low, mid - 1);
    } else {
```

Closed DAA/Screenshot 2025-11-03 at 21.09.32.pdf at 1cb41deb11db6e

Undo

```c
    printf("%d\n", numZeroes);
    return 0;
}
```