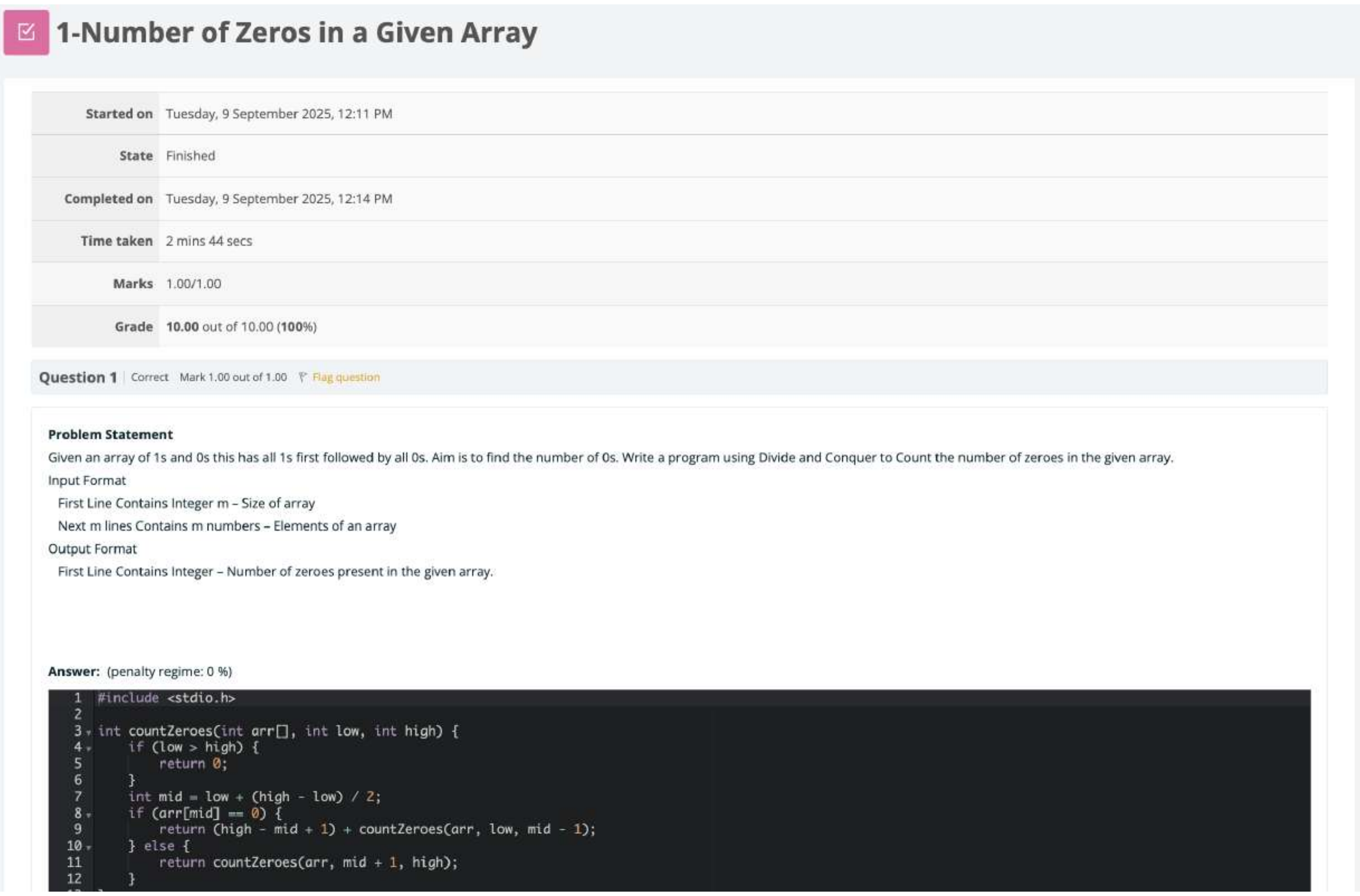


# 1-Number of Zeros in a Given Array

| | |
|---|---|
| **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 array.

Input Format

First Line Contains Integer m – Size of array

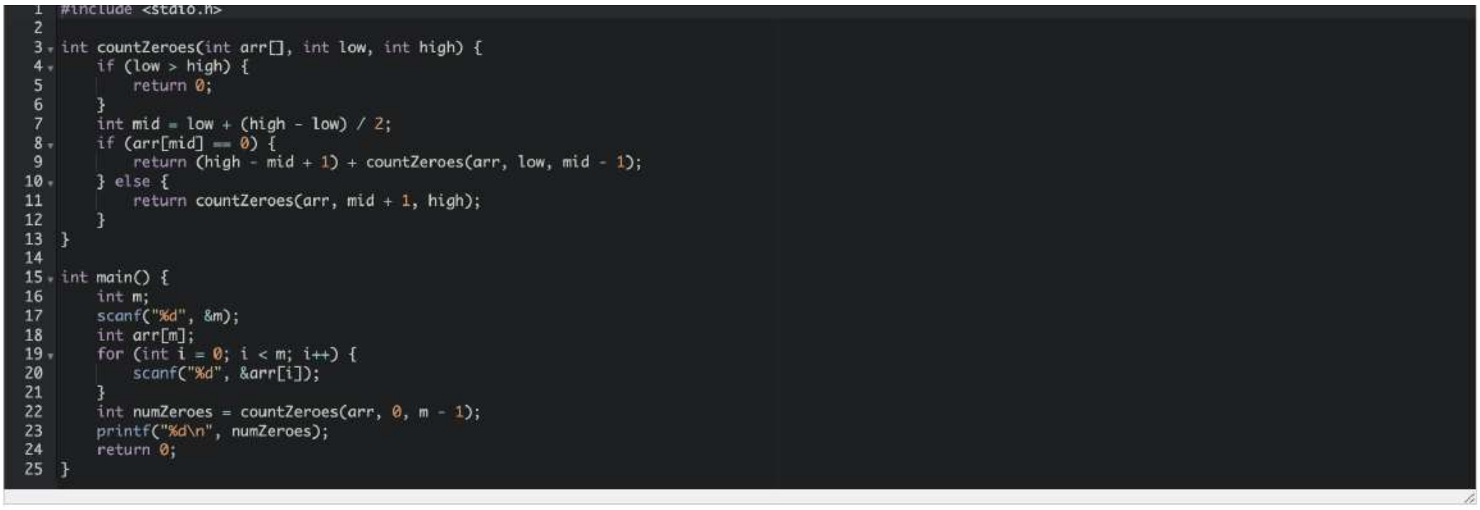
Next m lines Contains m numbers – Elements of an array

Output Format

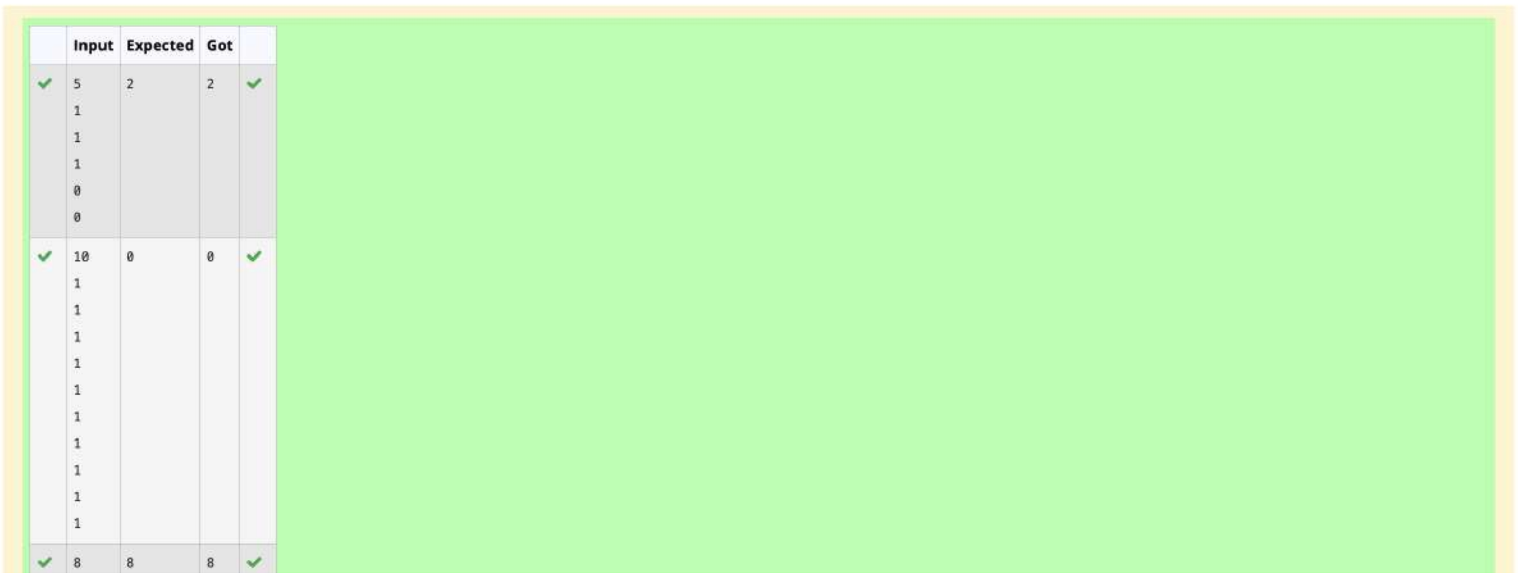
First Line Contains Integer – Number of zeroes present in the given array.

**Answer:** (penalty regime: 0 %)

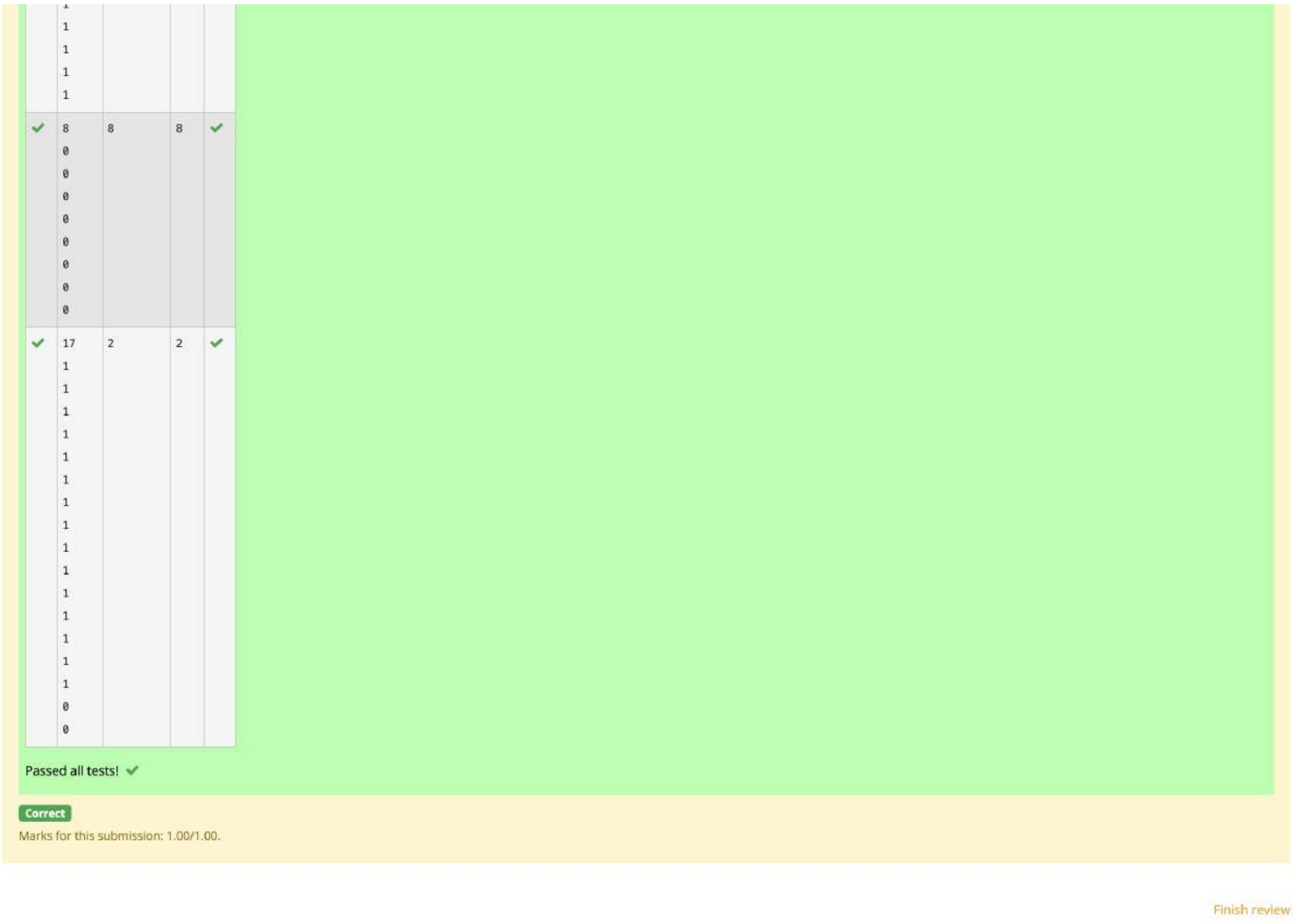
```
#include <stdio.h>

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 {
        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<br>1<br>1<br>1<br>0<br>0 | 2 | 2 | ✓ |
| ✓ | 10<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1 | 0 | 0 | ✓ |
| ✓ | 8 | 8 | 8 | ✓ |



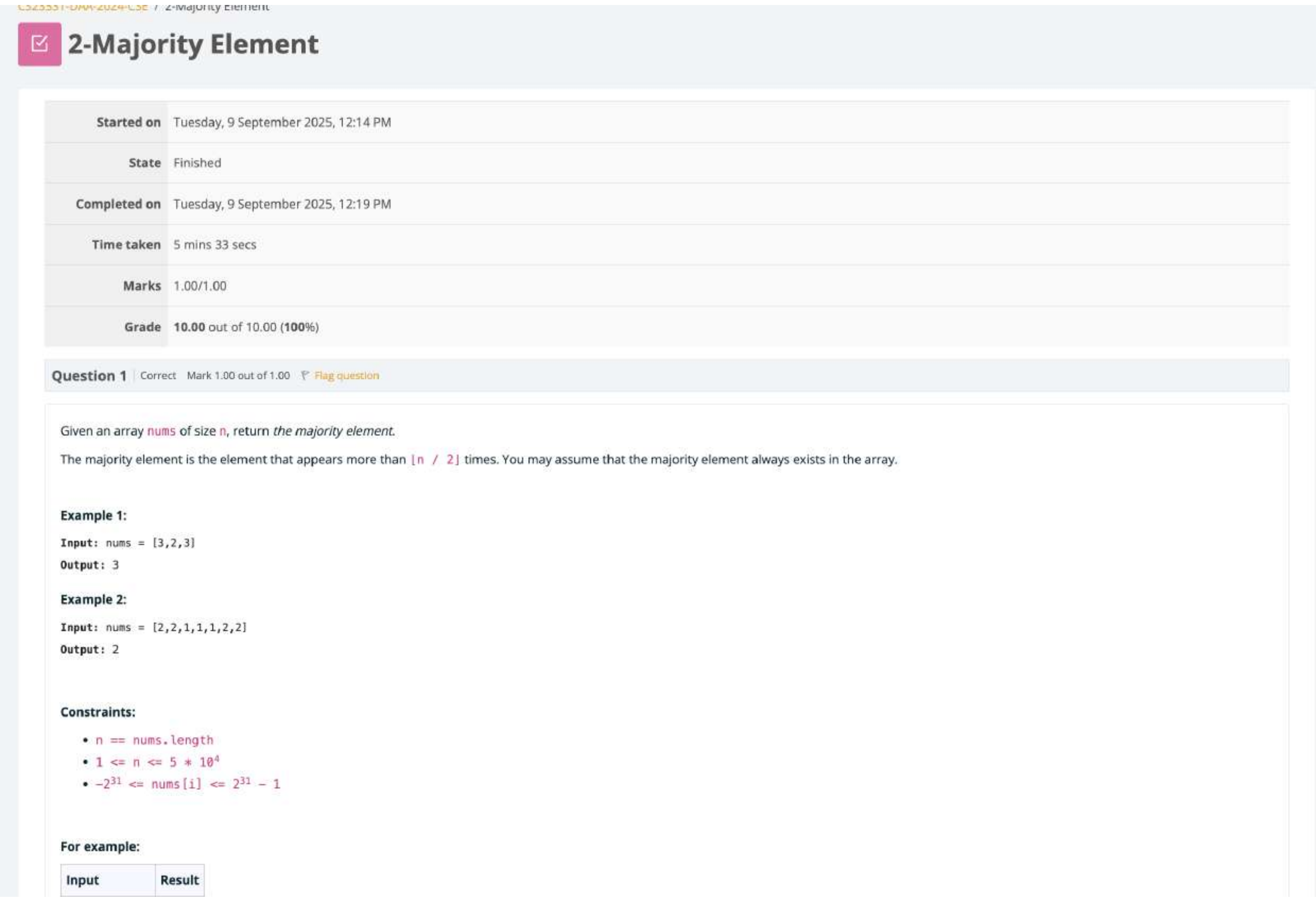
| | | | | |
|---|---|---|---|---|
| | 1<br>1<br>1<br>1 | | | |
| ✓ | 8<br>0<br>0<br>0<br>0<br>0<br>0<br>0<br>0 | 8 | 8 | ✓ |
| ✓ | 17<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>0<br>0 | 2 | 2 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review



# 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 33 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:**

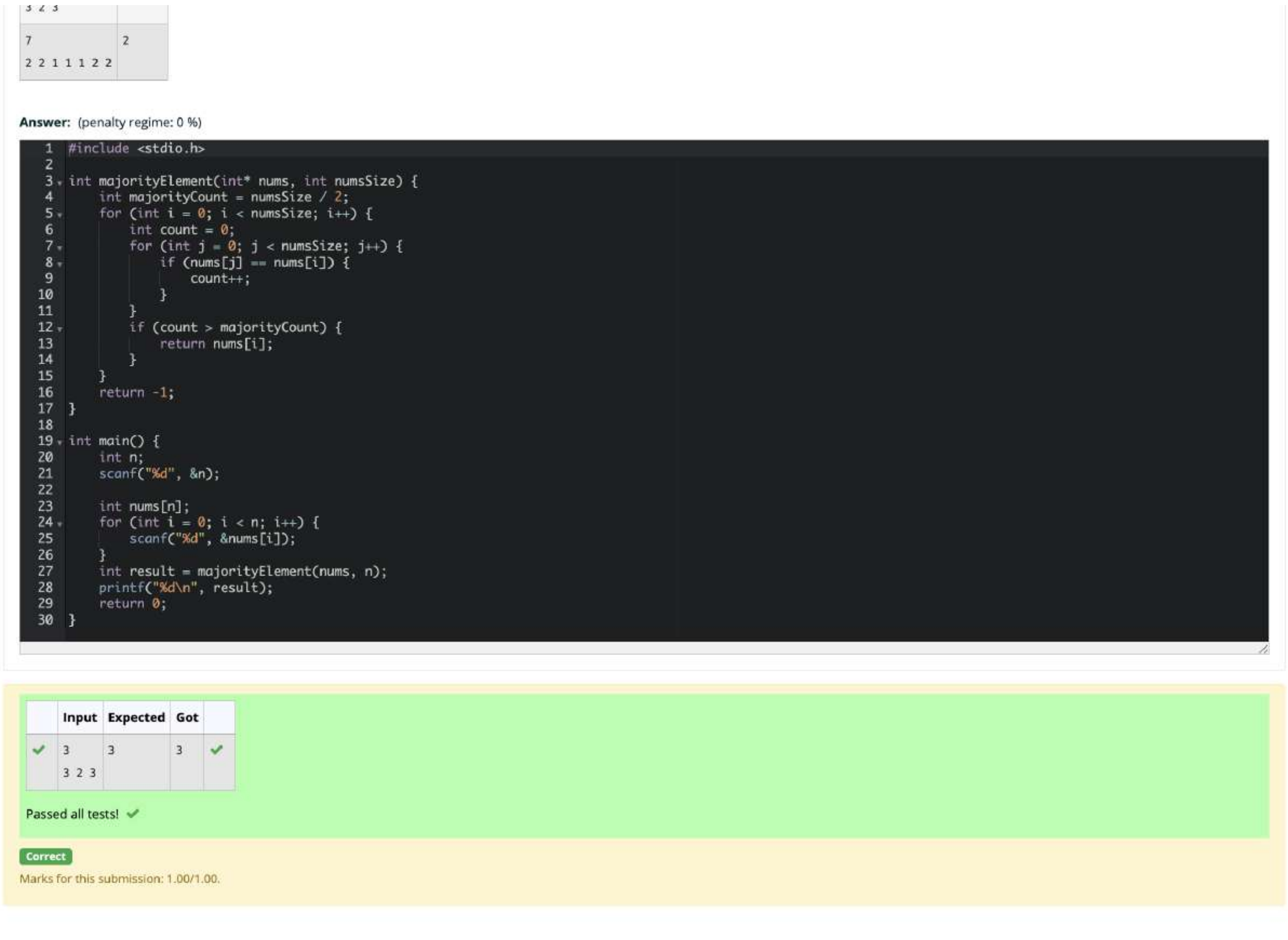
**Input:** nums = [2,2,1,1,1,2,2]
**Output:** 2

**Constraints:**

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

**For example:**

| Input | Result |
|---|---|



|  |  |
|---|---|
| 3 2 3 |  |
| 7<br>2 2 1 1 1 2 2 | 2 |

**Answer:** (penalty regime: 0 %)

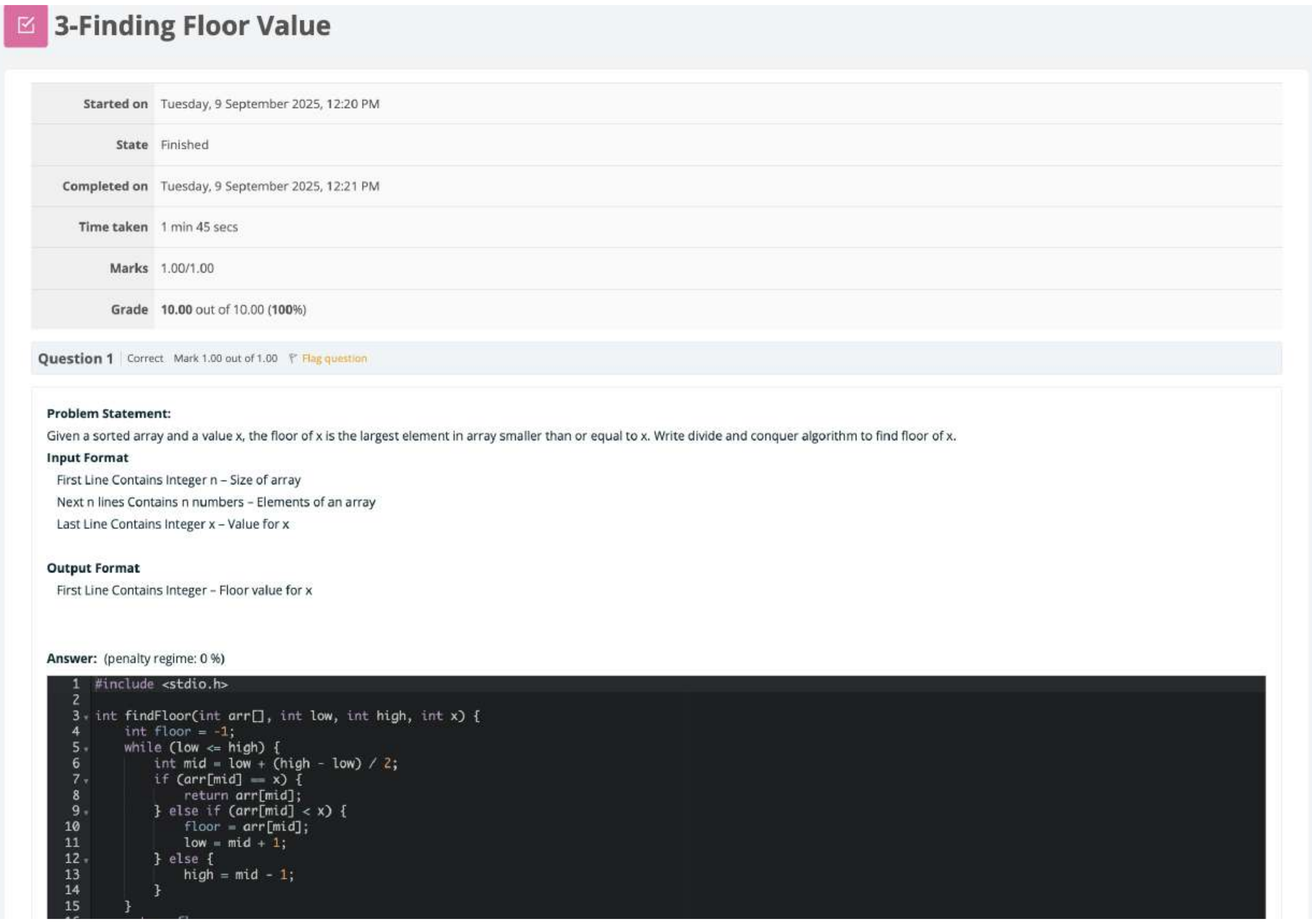
```c
#include <stdio.h>

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<br>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 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

**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

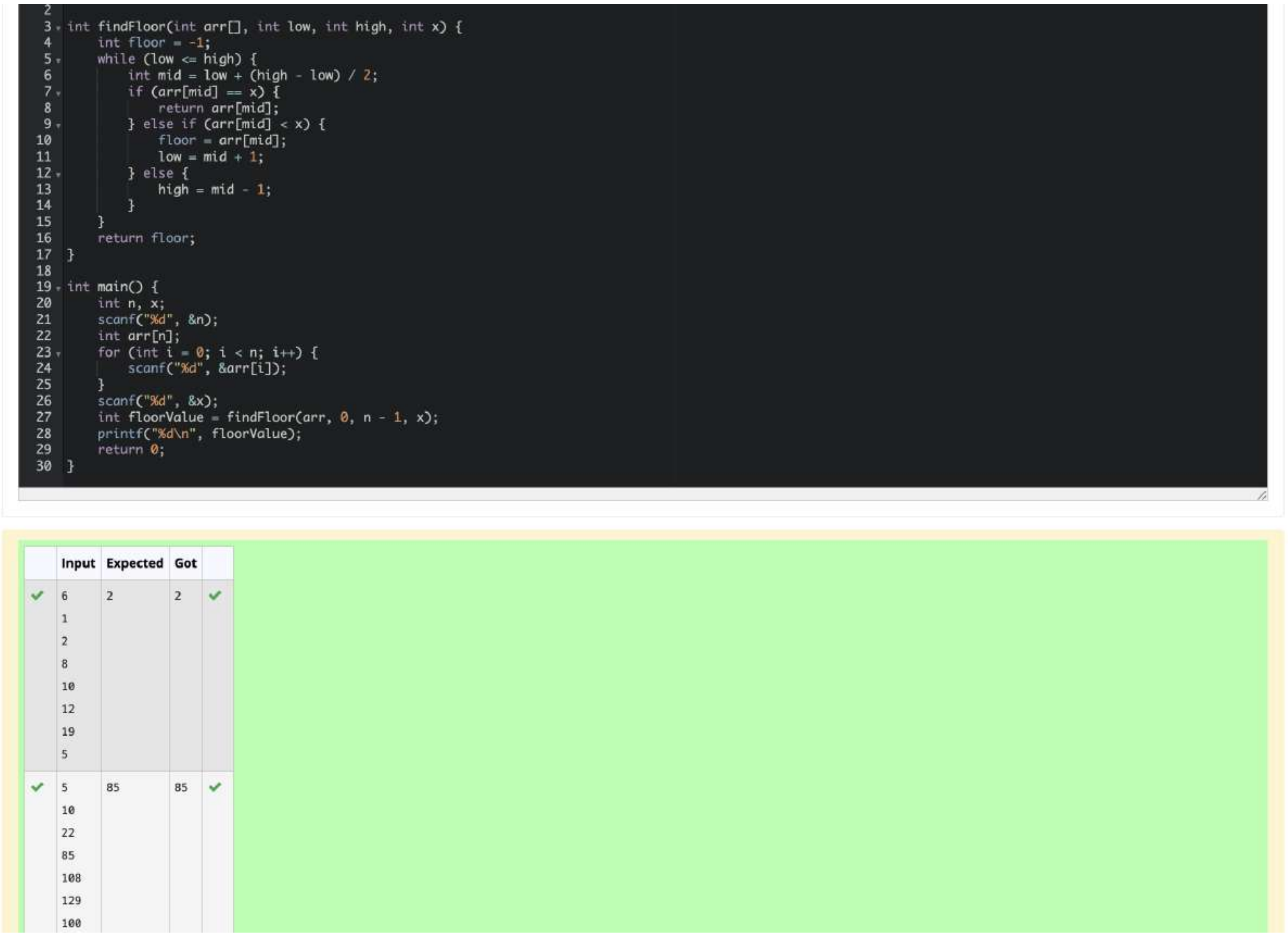
Last Line Contains Integer x – Value for x

**Output Format**

First Line Contains Integer – Floor value for x

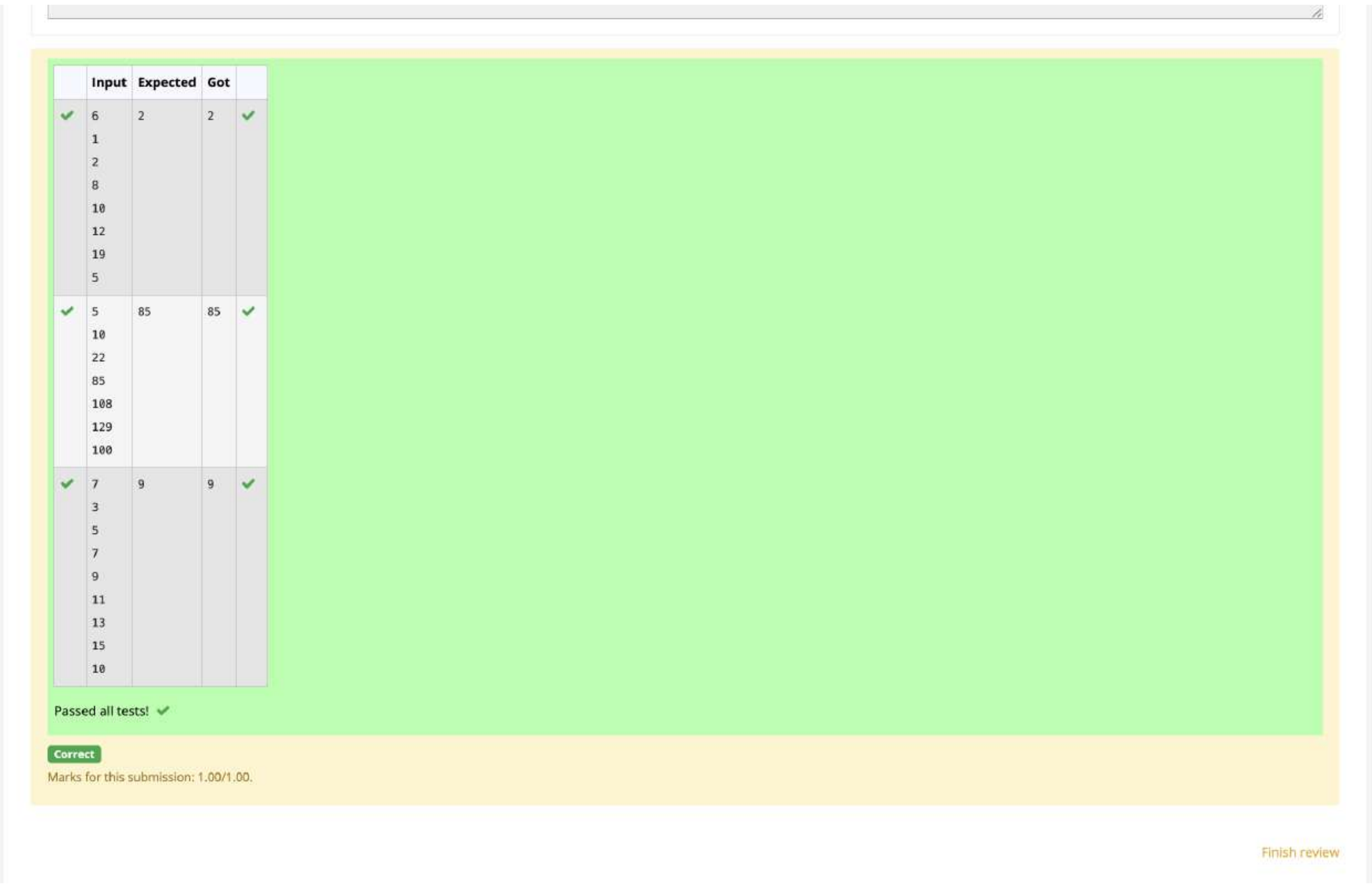
**Answer:** (penalty regime: 0 %)

```
#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);
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 6<br>1<br>2<br>8<br>10<br>12<br>19<br>5 | 2 | 2 | ✓ |
| ✓ | 5<br>10<br>22<br>85<br>108<br>129<br>100 | 85 | 85 | ✓ |



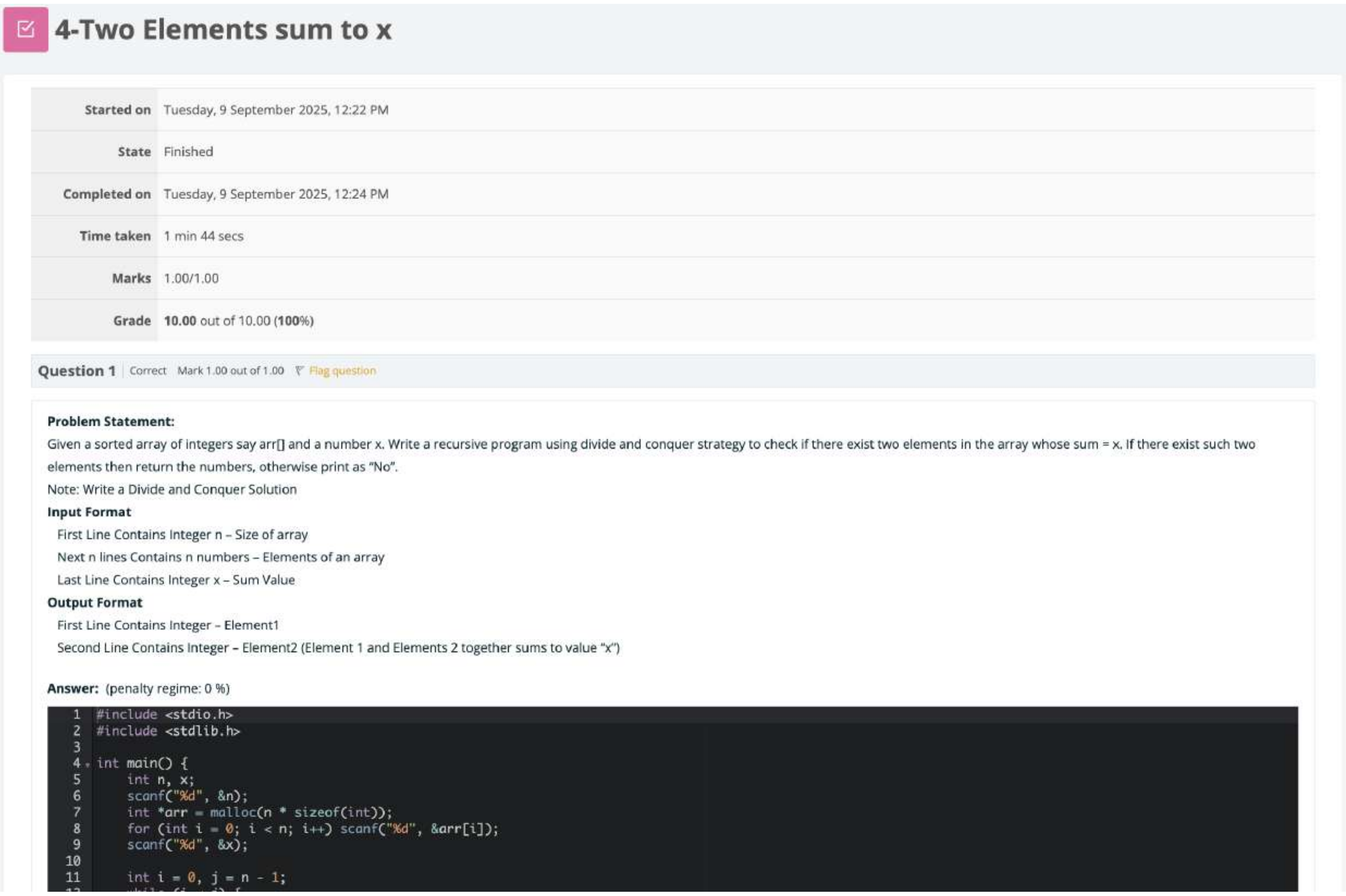
| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 6<br>1<br>2<br>8<br>10<br>12<br>19<br>5 | 2 | 2 | ✓ |
| ✓ | 5<br>10<br>22<br>85<br>108<br>129<br>100 | 85 | 85 | ✓ |
| ✓ | 7<br>3<br>5<br>7<br>9<br>11<br>13<br>15<br>10 | 9 | 9 | ✓ |

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

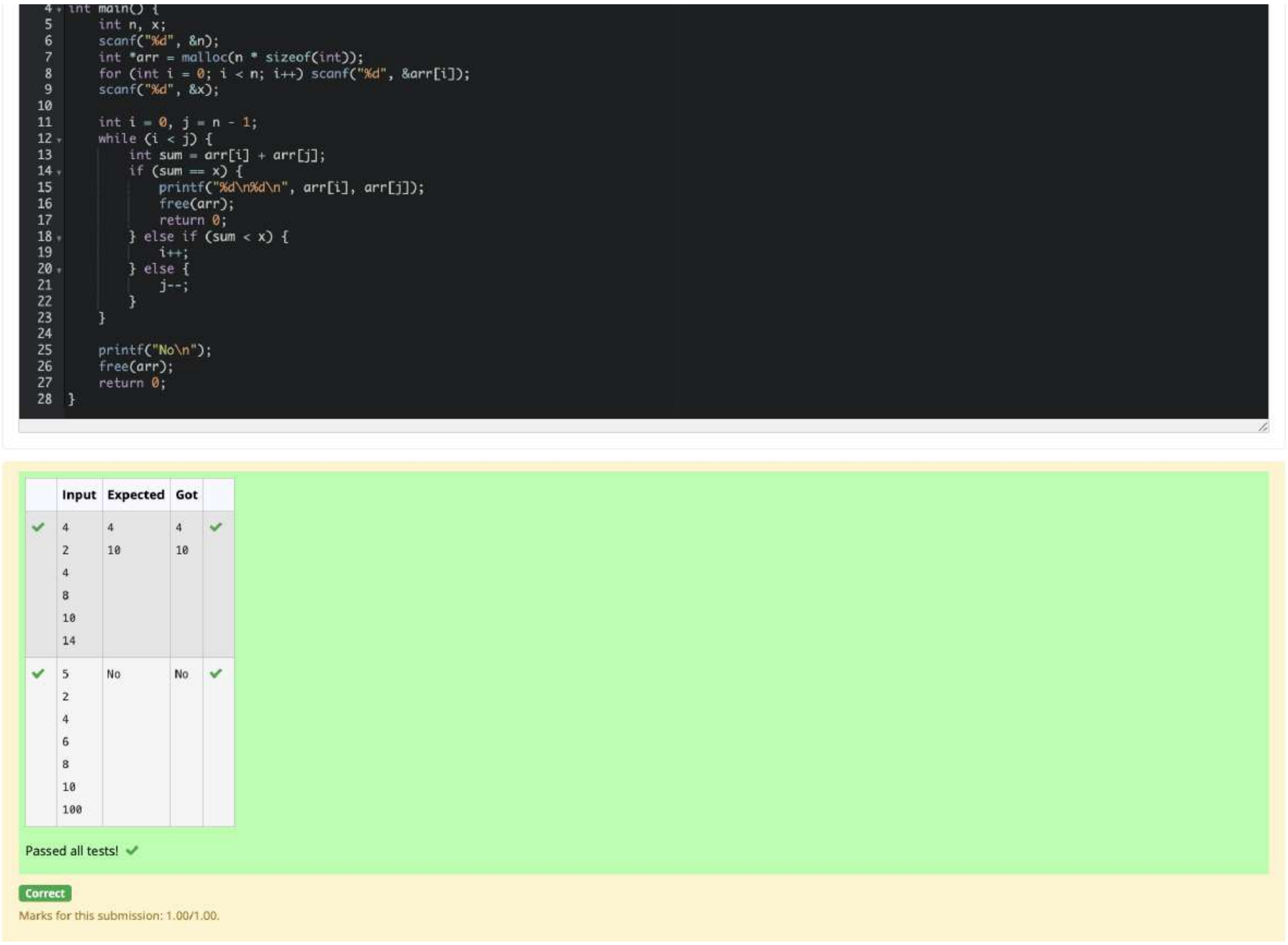
**Output Format**

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;
```



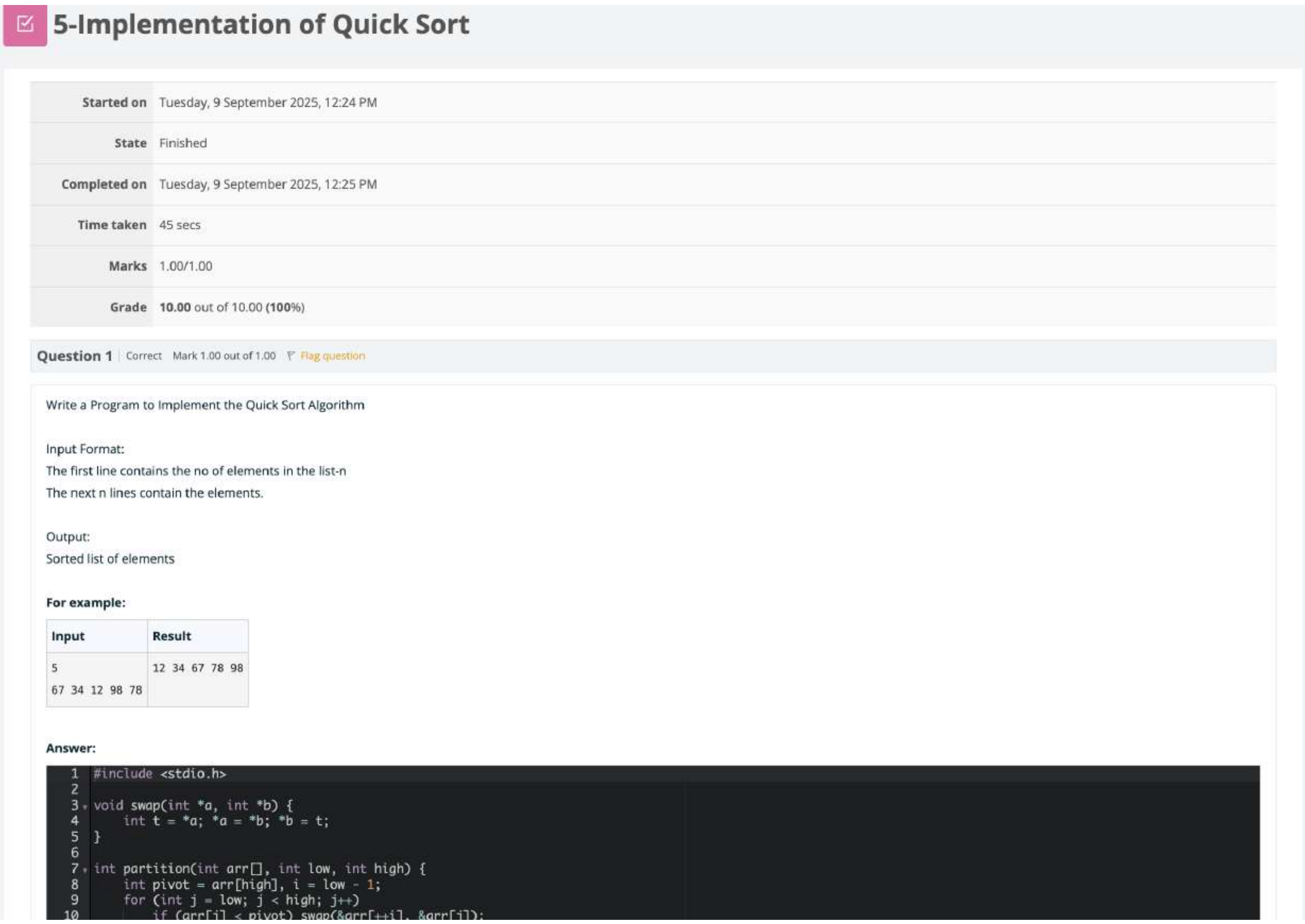
```c
int main() {
    int n, x;
    scanf("%d", &n);
    int *arr = malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) scanf("%d", &arr[i]);
    scanf("%d", &x);

    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<br>2<br>4<br>8<br>10<br>14 | 4<br>10 | 4<br>10 | ✓ |
| ✓ | 5<br>2<br>4<br>6<br>8<br>10<br>100 | No | No | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



# 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.

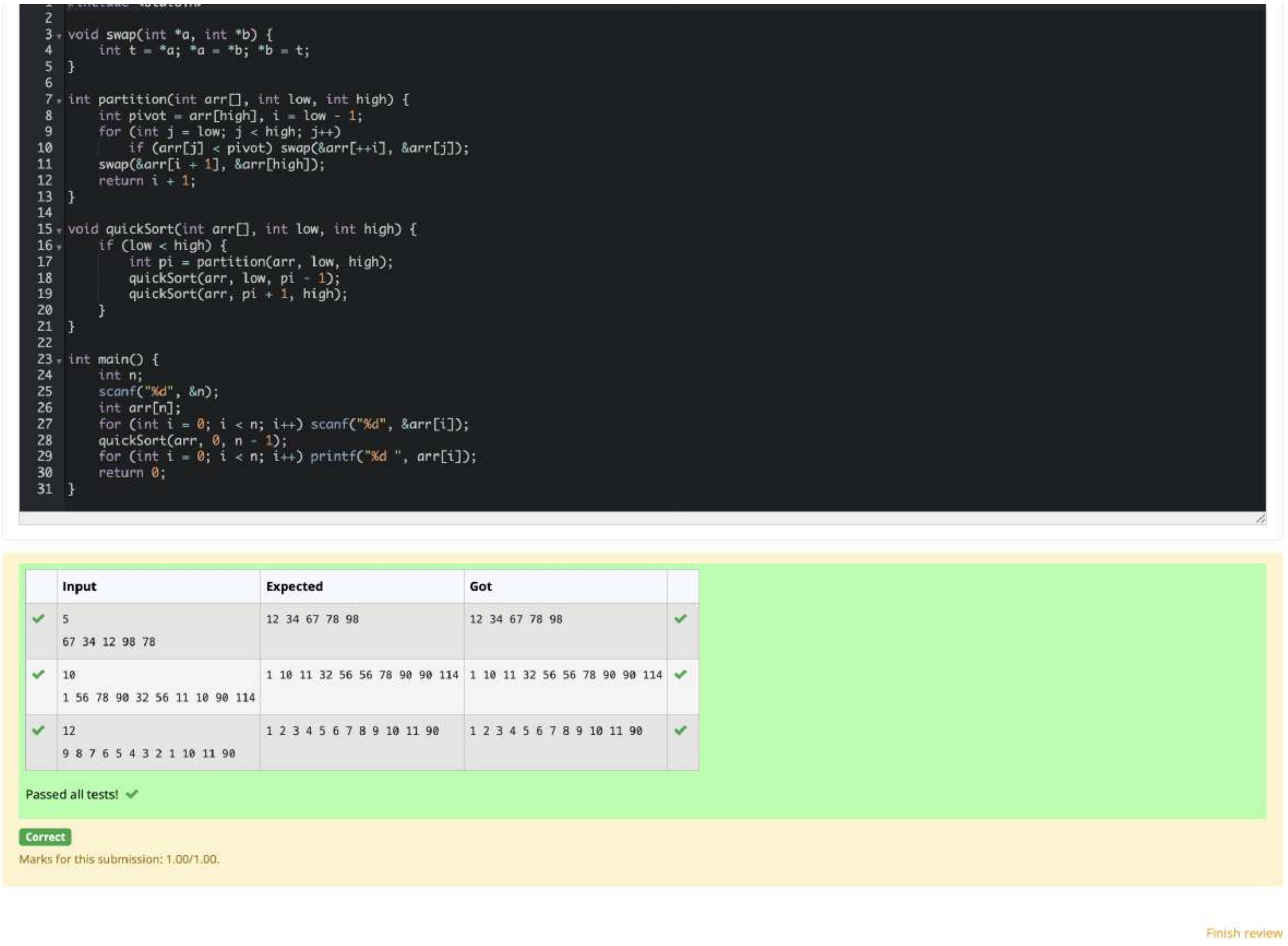
Output:
Sorted list of elements

**For example:**

| Input | Result |
|---|---|
| 5<br>67 34 12 98 78 | 12 34 67 78 98 |

**Answer:**

```
#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++)
        if (arr[j] < pivot) swap(&arr[++i], &arr[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;
}
```

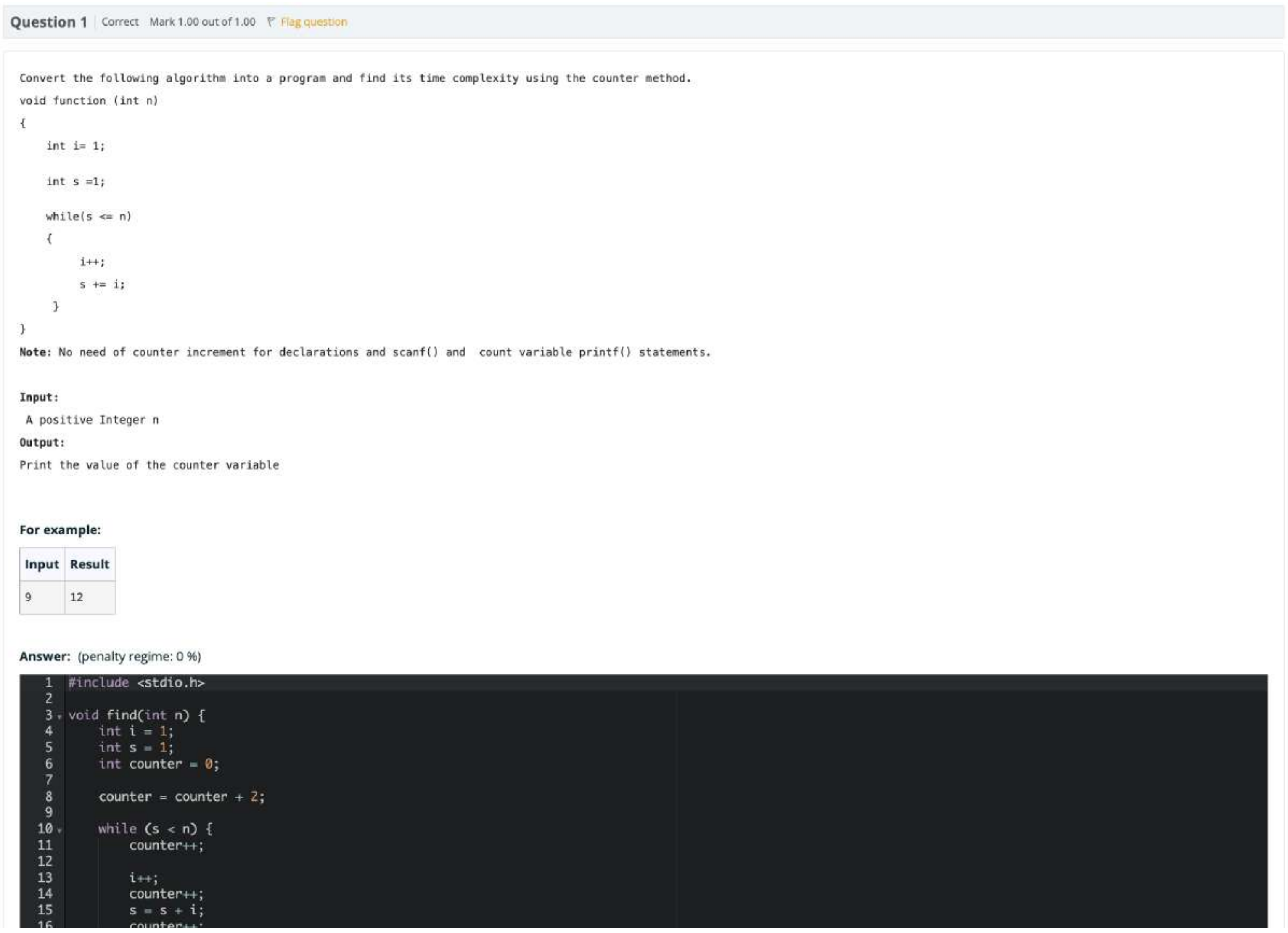
| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 5 67 34 12 98 78 | 12 34 67 78 98 | 12 34 67 78 98 | ✓ |
| ✓ | 10 1 56 78 90 32 56 11 10 90 114 | 1 10 11 32 56 56 78 90 90 114 | 1 10 11 32 56 56 78 90 90 114 | ✓ |
| ✓ | 12 9 8 7 6 5 4 3 2 1 10 11 90 | 1 2 3 4 5 6 7 8 9 10 11 90 | 1 2 3 4 5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

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review



**Question 1** Correct Mark 1.00 out of 1.00 Flag question

Convert the following algorithm into a program and find its time complexity using the counter method.

```
void function (int n)
{
    int i= 1;

    int s =1;

    while(s <= n)
    {
         i++;
         s += i;
     }
}
```

**Note:** No need of counter increment for declarations and scanf() and  count variable printf() statements.

**Input:**

A positive Integer n

**Output:**

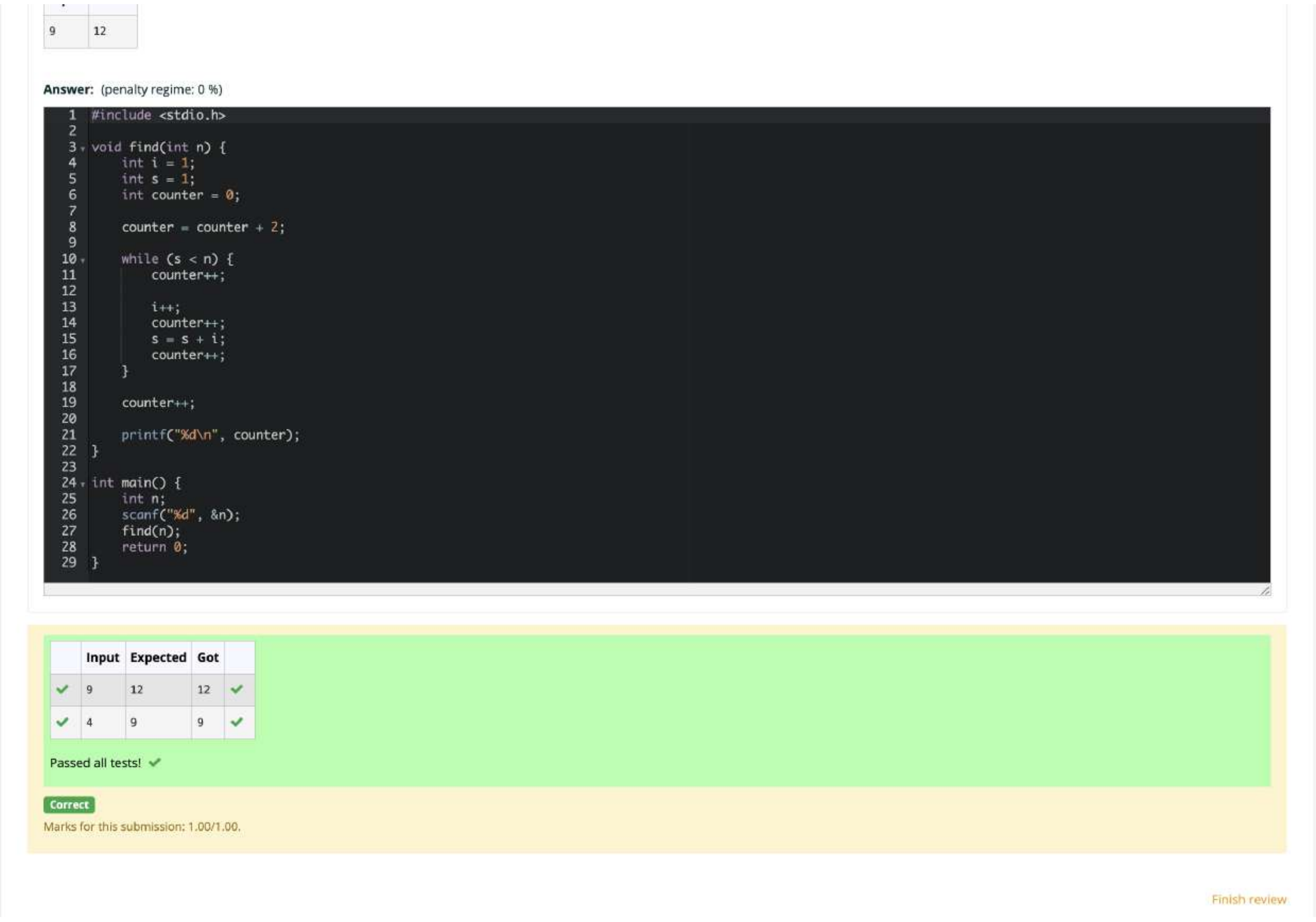
Print the value of the counter variable

**For example:**

| Input | Result |
|---|---|
| 9 | 12 |

**Answer:** (penalty regime: 0 %)

```
#include <stdio.h>

void find(int n) {
    int i = 1;
    int s = 1;
    int counter = 0;

    counter = counter + 2;

    while (s < n) {
        counter++;

        i++;
        counter++;
        s = s + i;
        counter++;
```



| | |
|---|---|
| 9 | 12 |

**Answer:** (penalty regime: 0 %)

```c
#include <stdio.h>

void find(int n) {
    int i = 1;
    int s = 1;
    int counter = 0;

    counter = counter + 2;

    while (s < n) {
        counter++;

        i++;
        counter++;
        s = s + i;
        counter++;
    }

    counter++;

    printf("%d\n", counter);
}

int main() {
    int n;
    scanf("%d", &n);
    find(n);
    return 0;
}
```

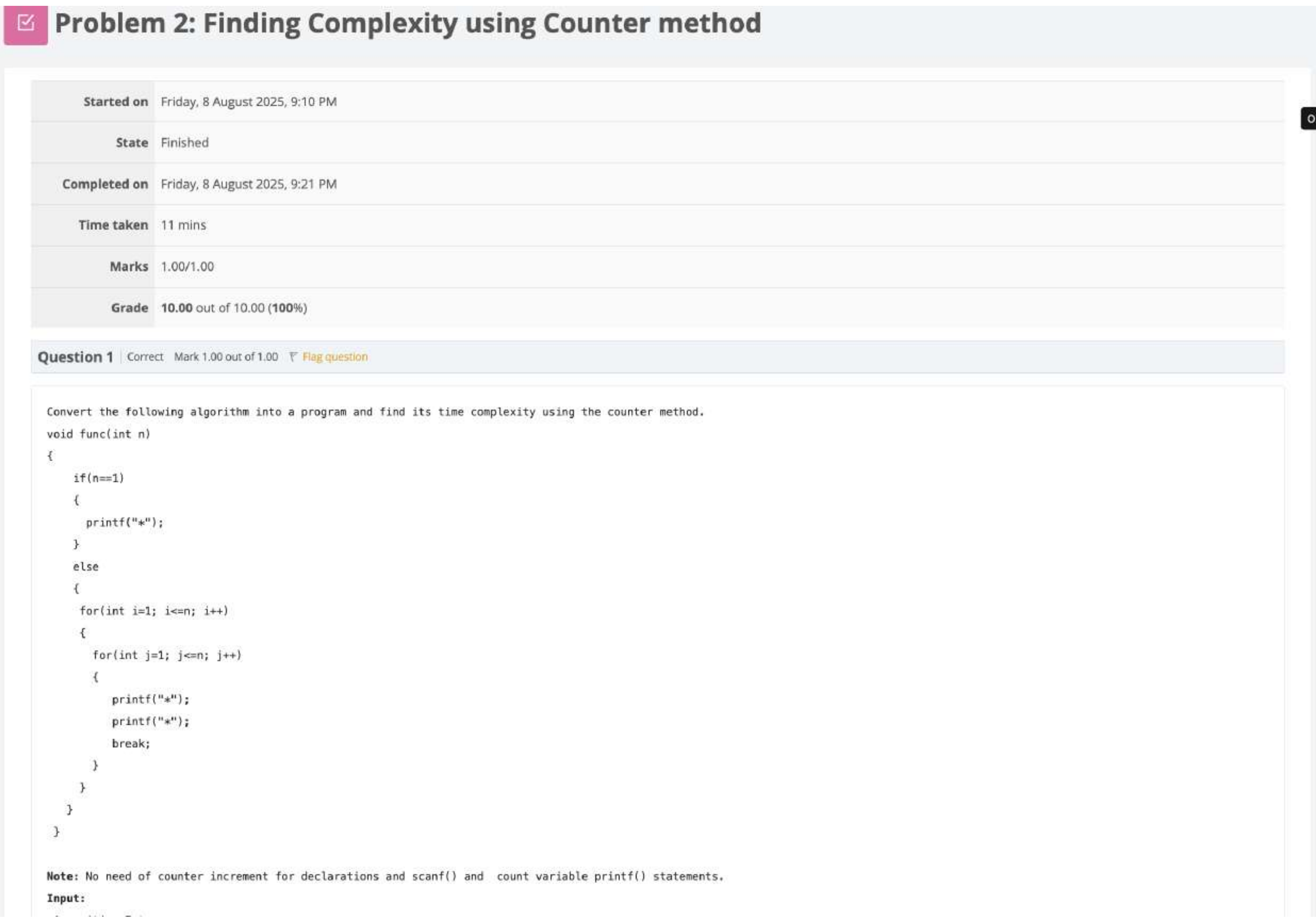
| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 9 | 12 | 12 | ✓ |
| ✓ | 4 | 9 | 9 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review



# Problem 2: Finding Complexity using Counter method

| | |
|---|---|
| **Started on** | Friday, 8 August 2025, 9:10 PM |
| **State** | Finished |
| **Completed on** | Friday, 8 August 2025, 9:21 PM |
| **Time taken** | 11 mins |
| **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

```
Convert the following algorithm into a program and find its time complexity using the counter method.
void func(int n)
{
    if(n==1)
    {
      printf("*");
    }
    else
    {
     for(int i=1; i<=n; i++)
     {
       for(int j=1; j<=n; j++)
       {
          printf("*");
          printf("*");
          break;
       }
     }
   }
 }
```

**Note:** No need of counter increment for declarations and scanf() and  count variable printf() statements.

**Input:**

```
A positive Integer n
```

**Output:**

```
Print the value of the counter variable
```

**Answer:** (penalty regime: 0 %)

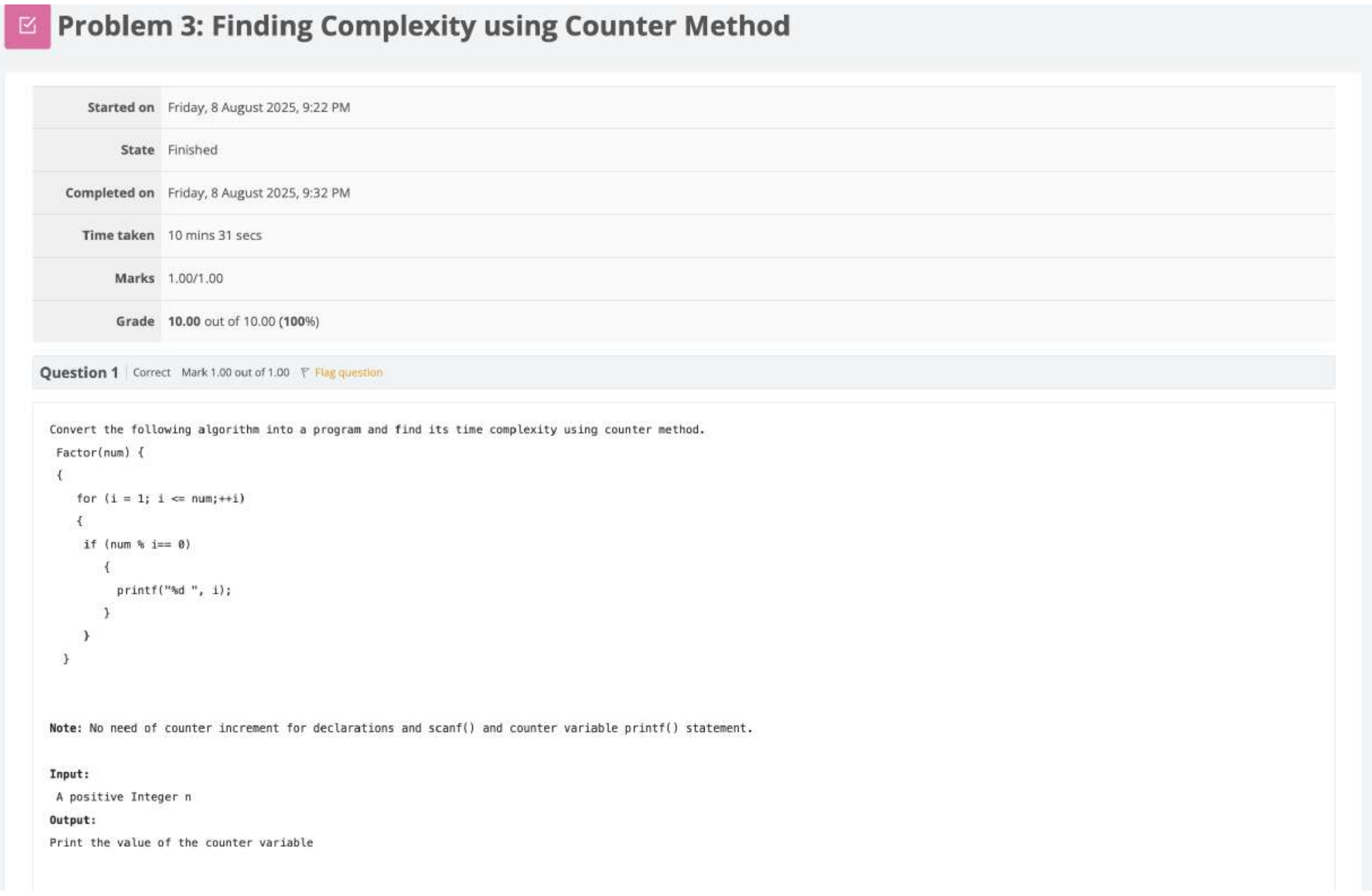
```c
#include <stdio.h>

void func(int n) {
    int counter = 0;

    counter = counter + 1;

    if (n == 1) {
    }
    else
    {
        counter = counter + 1;

        for (int i = 1; i <= n; i++) {
            counter = counter + 5;
        }
    }
    printf("%d\n", counter);
}

int main() {
    int n;
    scanf("%d", &n);
    func(n);
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 2 | 12 | 12 | ✓ |
| ✓ | 1000 | 5002 | 5002 | ✓ |
| ✓ | 143 | 717 | 717 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



# Problem 3: Finding Complexity using Counter Method

| | |
|---|---|
| **Started on** | Friday, 8 August 2025, 9:22 PM |
| **State** | Finished |
| **Completed on** | Friday, 8 August 2025, 9:32 PM |
| **Time taken** | 10 mins 31 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

Convert the following algorithm into a program and find its time complexity using counter method.

```
 Factor(num) {
 {
    for (i = 1; i <= num;++i)
    {
     if (num % i== 0)
        {
          printf("%d ", i);
        }
     }
  }
```

**Note:** No need of counter increment for declarations and scanf() and counter variable printf() statement.

**Input:**

 A positive Integer n

**Output:**

Print the value of the counter variable

**Output:**

```
Print the value of the counter variable
```

**Answer:**

```c
#include <stdio.h>

void Factor(int num) {
    int counter = 0;
    for (int i = 1; i <= num; i++) {
        counter++;
        counter++;
        if (num % i == 0) {
            counter++;
        }
    }
    counter++;
    printf("%d\n", counter);
}

int main() {
    int n;
    scanf("%d", &n);
    Factor(n);
    return 0;
}

```

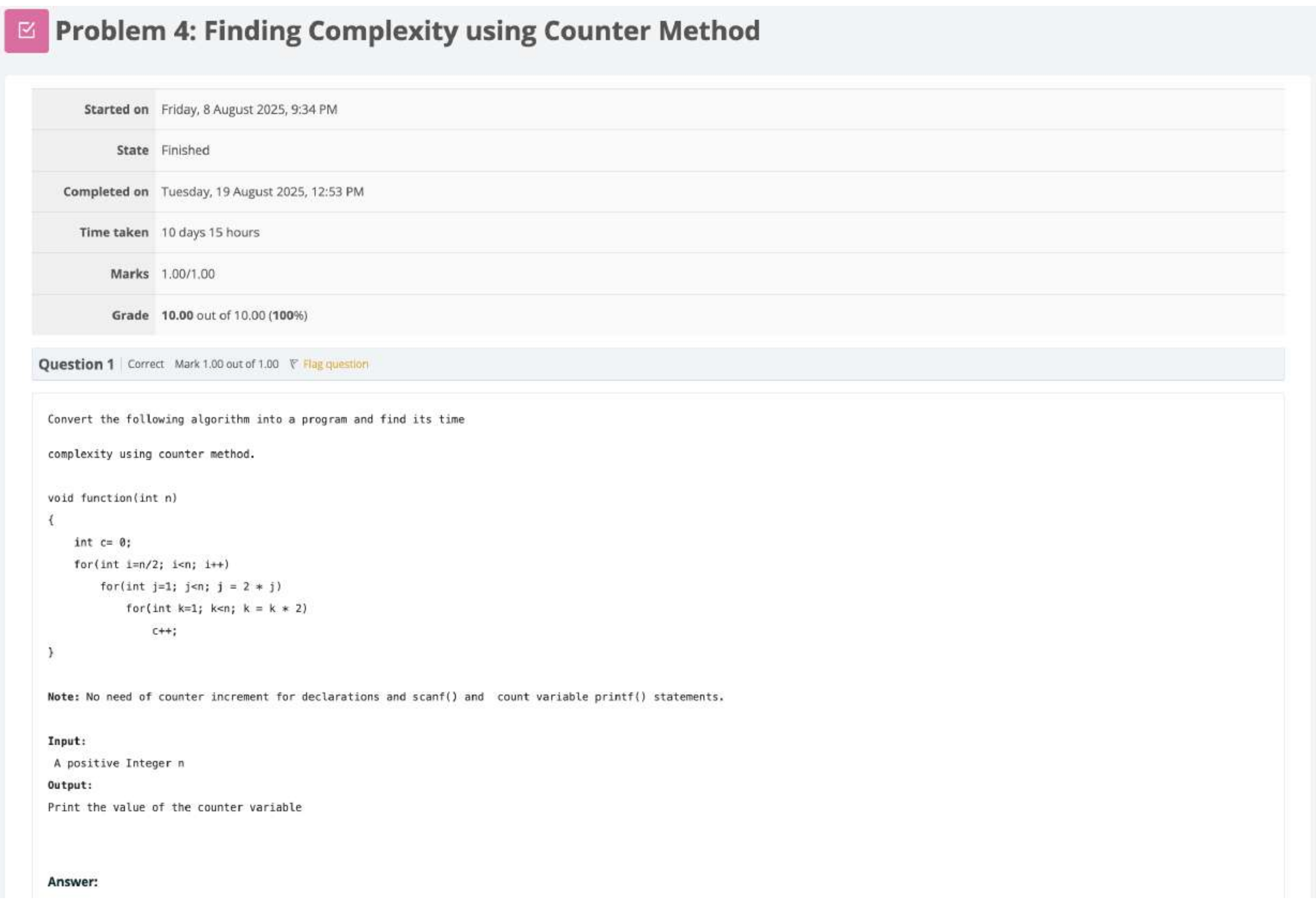
| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 12 | 31 | 31 | ✓ |
| ✓ | 25 | 54 | 54 | ✓ |
| ✓ | 4 | 12 | 12 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review



# Problem 4: Finding Complexity using Counter Method

| | |
|---|---|
| **Started on** | Friday, 8 August 2025, 9:34 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 19 August 2025, 12:53 PM |
| **Time taken** | 10 days 15 hours |
| **Marks** | 1.00/1.00 |
| **Grade** | **10.00** out of 10.00 (**100**%) |

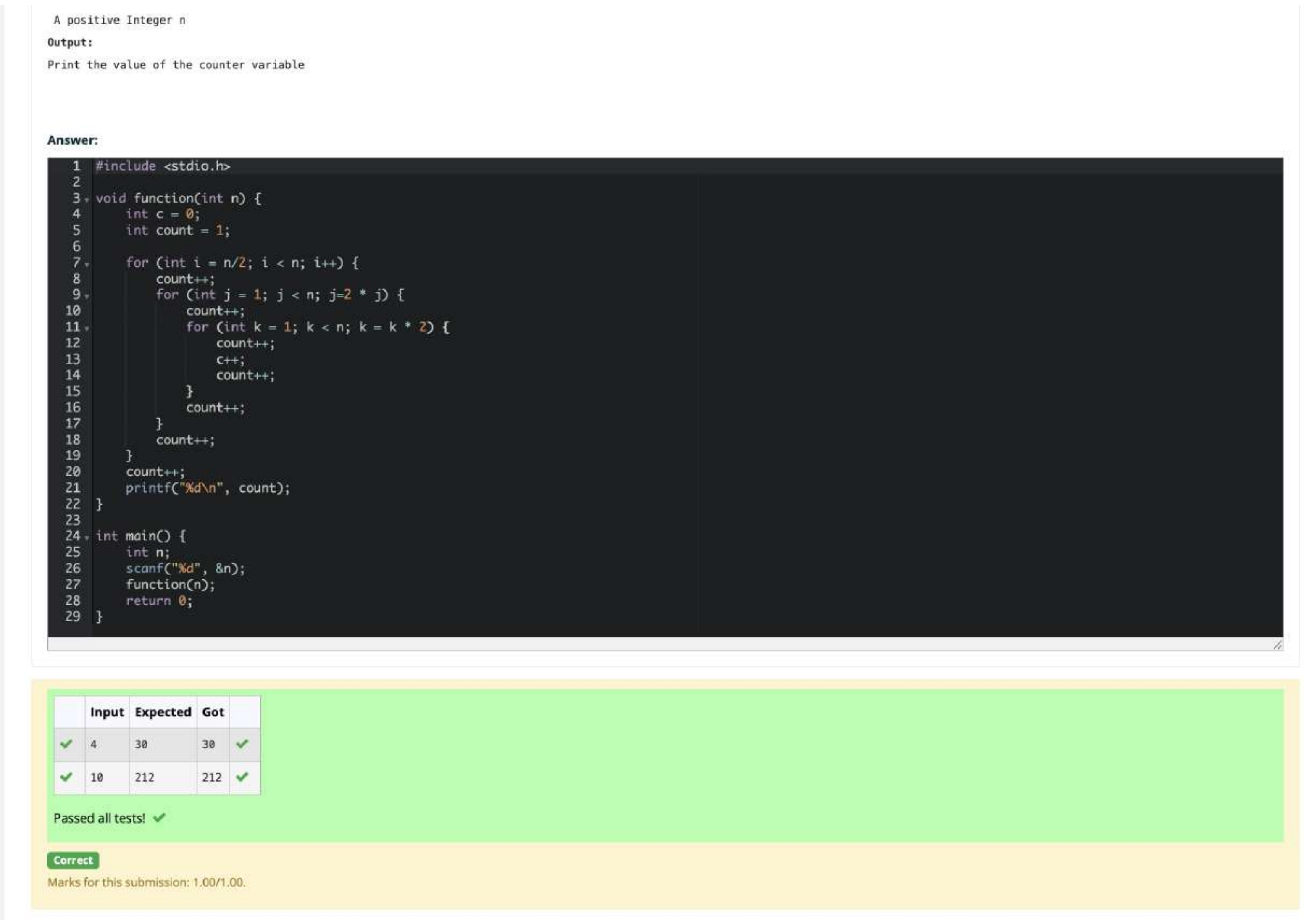
**Question 1** | Correct Mark 1.00 out of 1.00 Flag question

```
Convert the following algorithm into a program and find its time

complexity using counter method.

void function(int n)
{
    int c= 0;
    for(int i=n/2; i<n; i++)
        for(int j=1; j<n; j = 2 * j)
            for(int k=1; k<n; k = k * 2)
                c++;
}
```

**Note:** No need of counter increment for declarations and scanf() and  count variable printf() statements.

**Input:**
 A positive Integer n
**Output:**
Print the value of the counter variable

**Answer:**



```
A positive Integer n
```

**Output:**

```
Print the value of the counter variable
```

**Answer:**

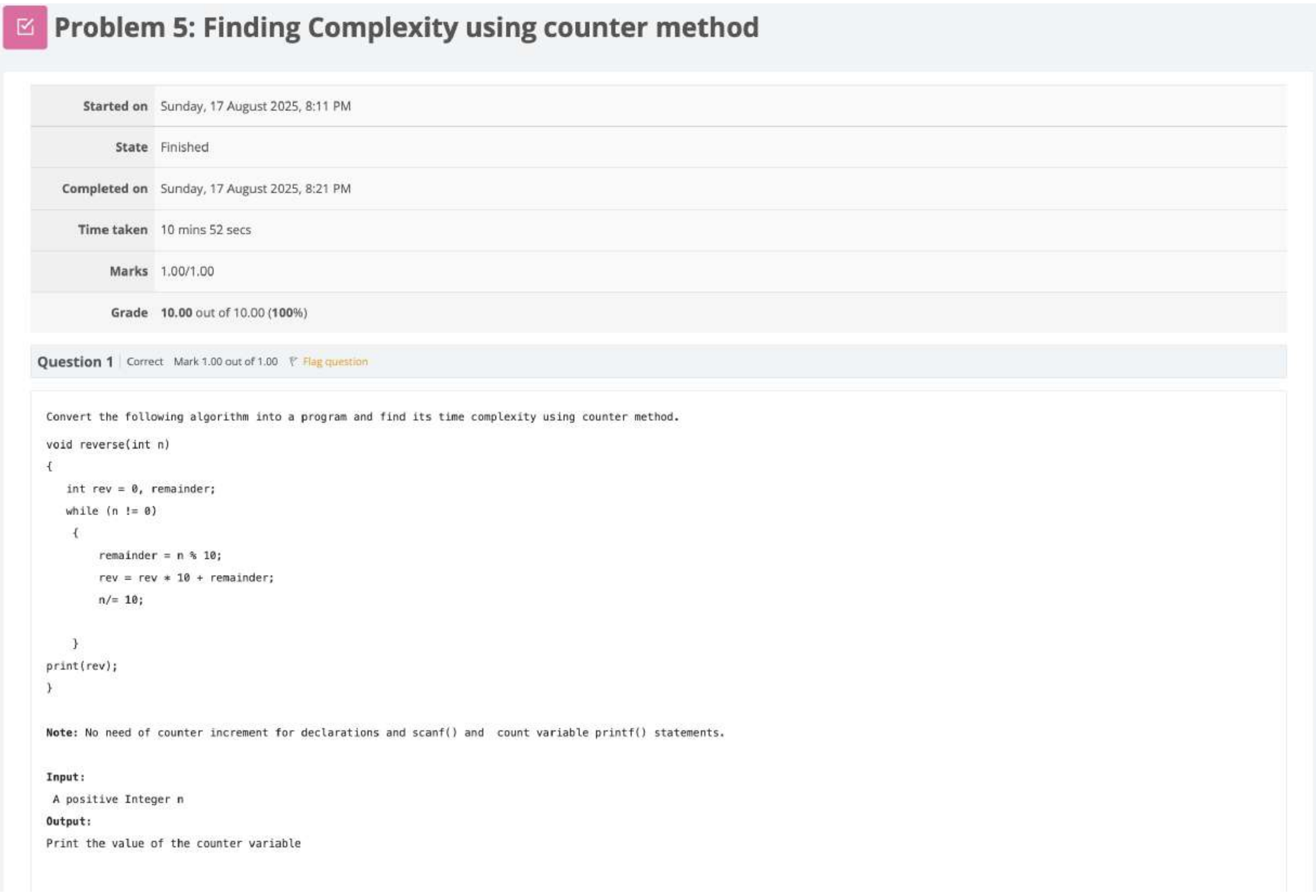
```c
#include <stdio.h>

void function(int n) {
    int c = 0;
    int count = 1;

    for (int i = n/2; i < n; i++) {
        count++;
        for (int j = 1; j < n; j=2 * j) {
            count++;
            for (int k = 1; k < n; k = k * 2) {
                count++;
                c++;
                count++;
            }
            count++;
        }
        count++;
    }
    count++;
    printf("%d\n", count);
}

int main() {
    int n;
    scanf("%d", &n);
    function(n);
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 4 | 30 | 30 | ✓ |
| ✓ | 10 | 212 | 212 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



## Problem 5: Finding Complexity using counter method

| | |
|---|---|
| **Started on** | Sunday, 17 August 2025, 8:11 PM |
| **State** | Finished |
| **Completed on** | Sunday, 17 August 2025, 8:21 PM |
| **Time taken** | 10 mins 52 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

Convert the following algorithm into a program and find its time complexity using counter method.

```
void reverse(int n)
{
   int rev = 0, remainder;
   while (n != 0)
    {
        remainder = n % 10;
        rev = rev * 10 + remainder;
        n/= 10;

    }
print(rev);
}
```

**Note:** No need of counter increment for declarations and scanf() and  count variable printf() statements.

**Input:**
 A positive Integer n
**Output:**
Print the value of the counter variable

**Input:**

A positive Integer n

**Output:**

Print the value of the counter variable

**Answer:**

```c
#include <stdio.h>

void reverse(int n) {
    int rev = 0, remainder;
    int count = 3;

    while (n != 0) {
        remainder = n % 10;
        count++;
        rev = rev * 10 + remainder;
        count += 2;
        n /= 10;
        count++;
    }

    printf("%d\n", count);
}

int main() {
    int n;
    scanf("%d", &n);
    reverse(n);
    return 0;
}
```

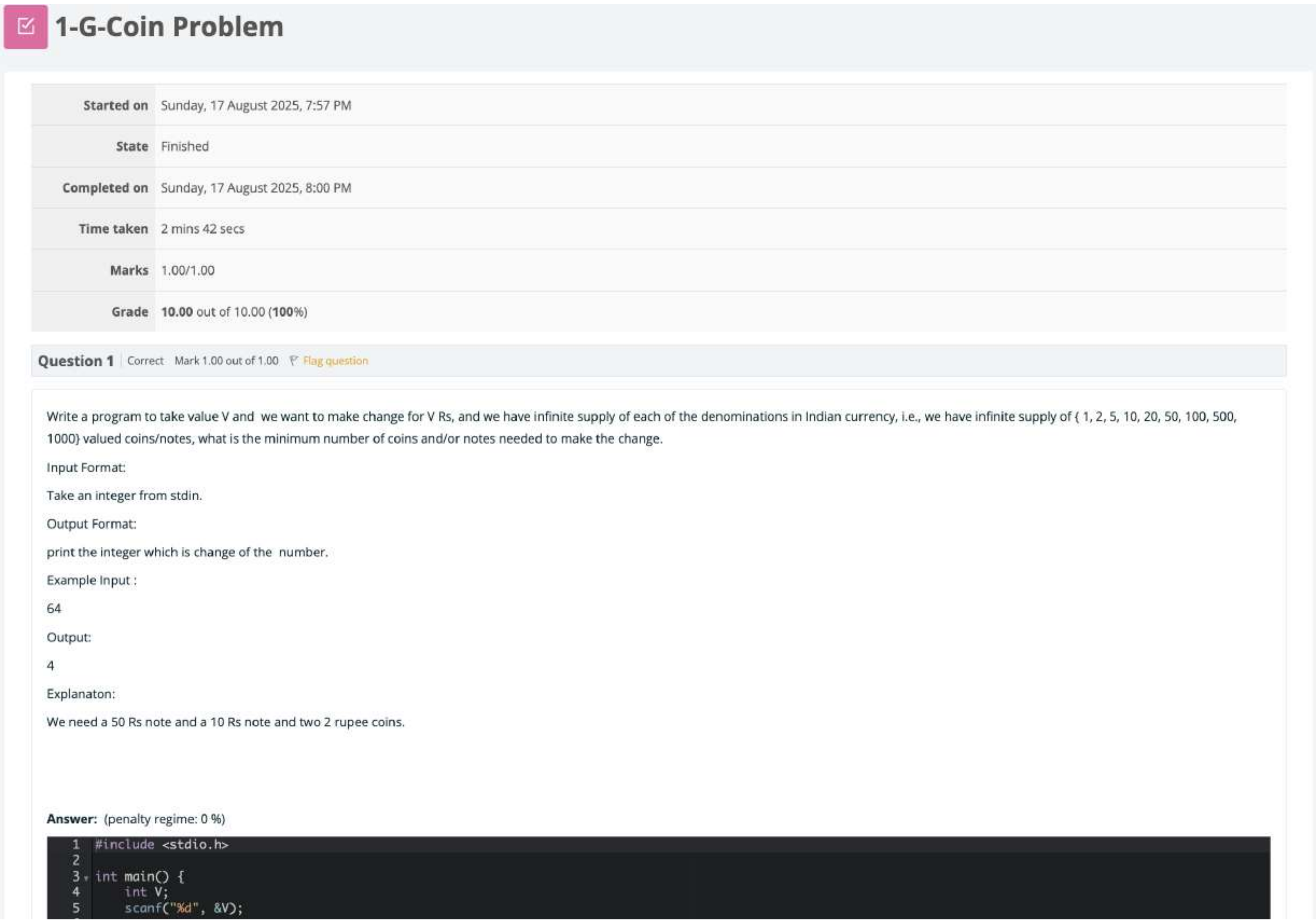
| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 12 | 11 | 11 | ✓ |
| ✓ | 1234 | 19 | 19 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review



# 1-G-Coin Problem

| | |
|---|---|
| **Started on** | Sunday, 17 August 2025, 7:57 PM |
| **State** | Finished |
| **Completed on** | Sunday, 17 August 2025, 8:00 PM |
| **Time taken** | 2 mins 42 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 take value V and we want to make change for V Rs, and we have infinite supply of each of the denominations in Indian currency, i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change.

Input Format:

Take an integer from stdin.

Output Format:

print the integer which is change of the number.

Example Input :

64

Output:

4

Explanaton:

We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.

**Answer:** (penalty regime: 0 %)

```
#include <stdio.h>

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

**Answer:** (penalty regime: 0 %)

```c
#include <stdio.h>

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

    int denominations[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
    int num_denominations = sizeof(denominations) / sizeof(denominations[0]);
    int count = 0;

    for (int i = 0; i < num_denominations; i++) {
        while (V >= denominations[i]) {
            V -= denominations[i];
            count++;
        }
    }

    printf("%d\n", count);

    return 0;
}
```

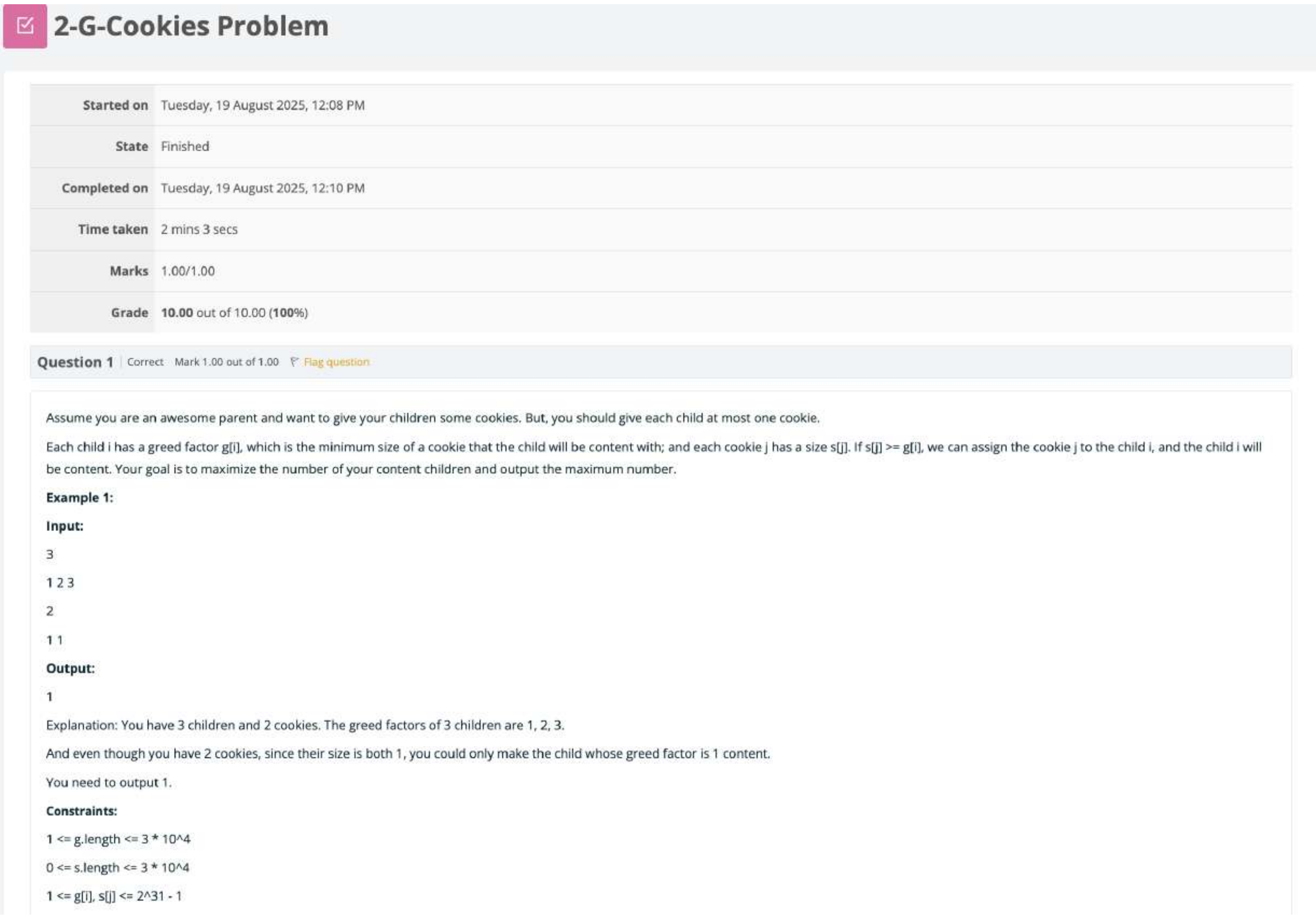
| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 49 | 5 | 5 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review



# 2-G-Cookies Problem

| | |
|---|---|
| **Started on** | Tuesday, 19 August 2025, 12:08 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 19 August 2025, 12:10 PM |
| **Time taken** | 2 mins 3 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

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.

Each child i has a greed factor g[i], which is the minimum size of a cookie that the child will be content with; and each cookie j has a size s[j]. If s[j] >= g[i], we can assign the cookie j to the child i, and the child i will be content. Your goal is to maximize the number of your content children and output the maximum number.

**Example 1:**

**Input:**

3

1 2 3

2

1 1

**Output:**

1

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.

And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.

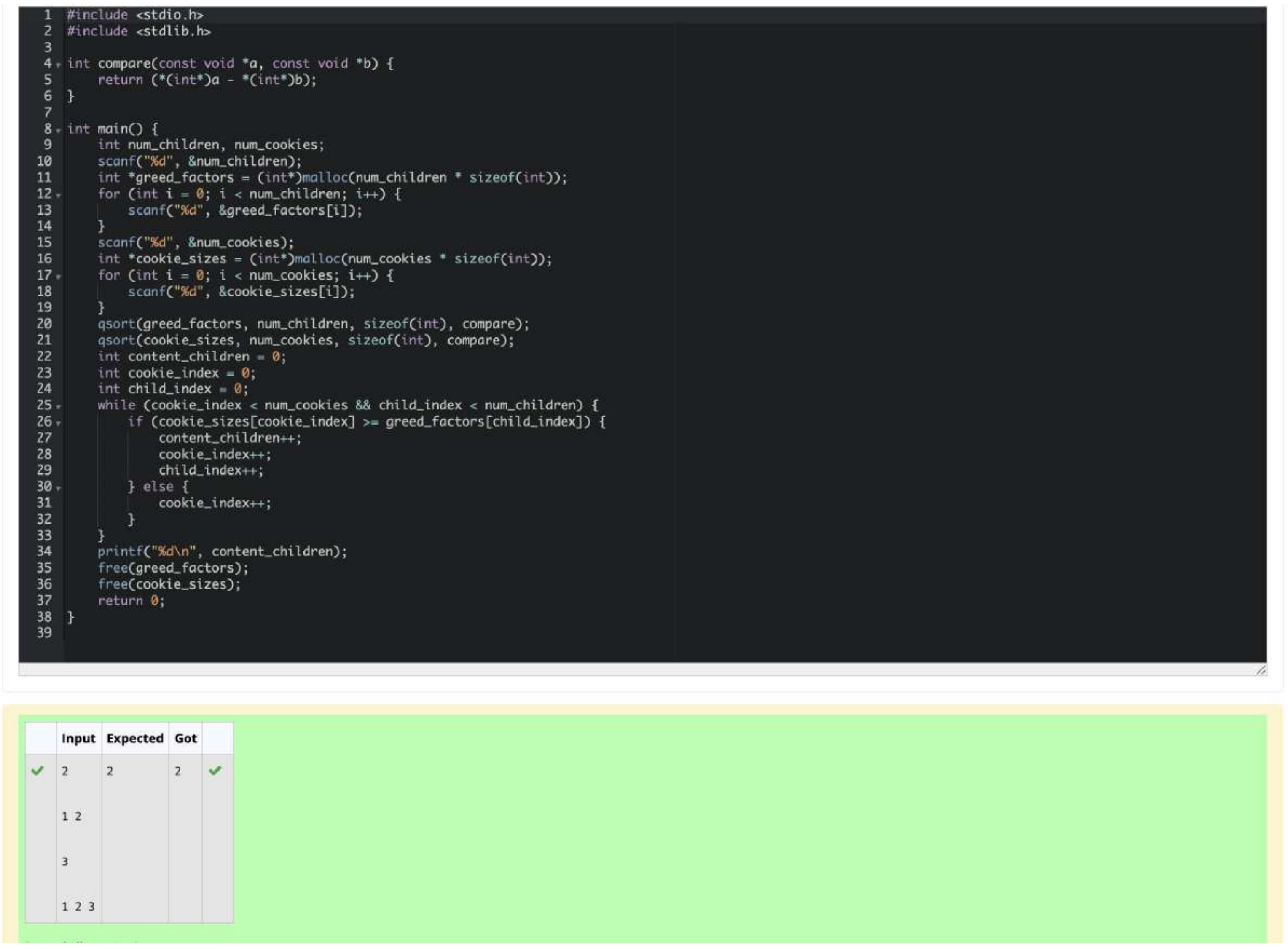
You need to output 1.

**Constraints:**

1 <= g.length <= 3 * 10^4

0 <= s.length <= 3 * 10^4

1 <= g[i], s[j] <= 2^31 - 1



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

int compare(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}

int main() {
    int num_children, num_cookies;
    scanf("%d", &num_children);
    int *greed_factors = (int*)malloc(num_children * sizeof(int));
    for (int i = 0; i < num_children; i++) {
        scanf("%d", &greed_factors[i]);
    }
    scanf("%d", &num_cookies);
    int *cookie_sizes = (int*)malloc(num_cookies * sizeof(int));
    for (int i = 0; i < num_cookies; i++) {
        scanf("%d", &cookie_sizes[i]);
    }
    qsort(greed_factors, num_children, sizeof(int), compare);
    qsort(cookie_sizes, num_cookies, sizeof(int), compare);
    int content_children = 0;
    int cookie_index = 0;
    int child_index = 0;
    while (cookie_index < num_cookies && child_index < num_children) {
        if (cookie_sizes[cookie_index] >= greed_factors[child_index]) {
            content_children++;
            cookie_index++;
            child_index++;
        } else {
            cookie_index++;
        }
    }
    printf("%d\n", content_children);
    free(greed_factors);
    free(cookie_sizes);
    return 0;
}

```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 2<br>1 2<br>3<br>1 2 3 | 2 | 2 | ✓ |

# 3-G-Burger Problem

| | |
|---|---|
| **Started on** | Sunday, 24 August 2025, 6:02 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 26 August 2025, 12:58 PM |
| **Time taken** | 1 day 18 hours |
| **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

A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person needs to run a distance to burn out his calories. If he has eaten $i$ burgers with c calories each, then he has to run at least $3^i * c$ kilometers to burn out the calories. For example, if he ate 3 burgers with the count of calorie in the order: [1, 3, 2], the kilometers he needs to run are $(3^0 * 1) + (3^1 * 3) + (3^2 * 2) = 1 + 9 + 18 = 28$. But this is not the minimum, so need to try out other orders of consumption and choose the minimum value. Determine the minimum distance he needs to run. Note: He can eat burger in any order and use an efficient sorting algorithm.Apply greedy approach to solve the problem.

**Input Format**

First Line contains the number of burgers

Second line contains calories of each burger which is n space-separate integers

**Output Format**

Print: Minimum number of kilometers needed to run to burn out the calories

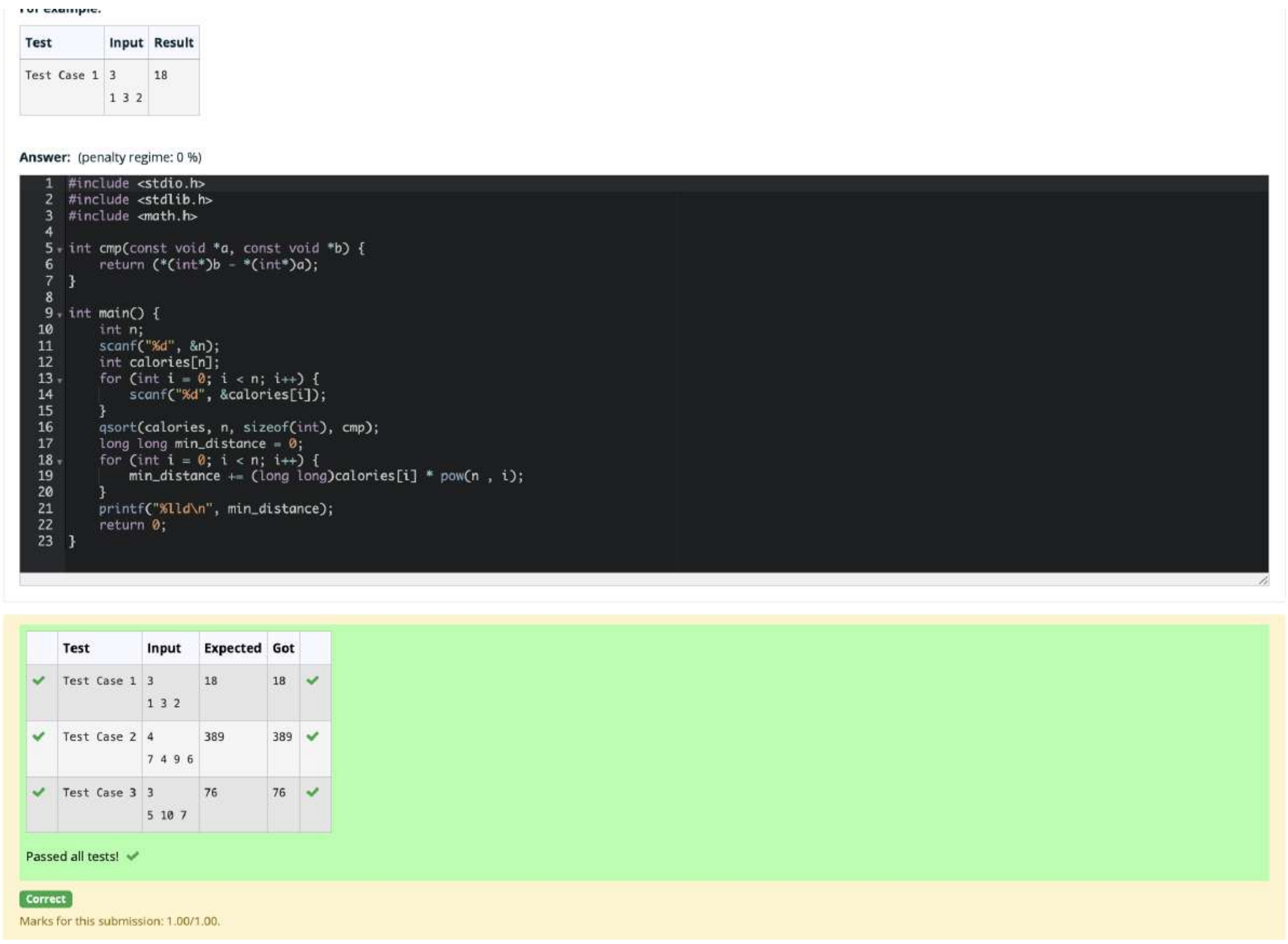
**Sample Input**

```
3
5 10 7
```

**Sample Output**

```
76
```

**For example:**


For example:

| Test | Input | Result |
|---|---|---|
| Test Case 1 | 3<br>1 3 2 | 18 |

**Answer:** (penalty regime: 0 %)

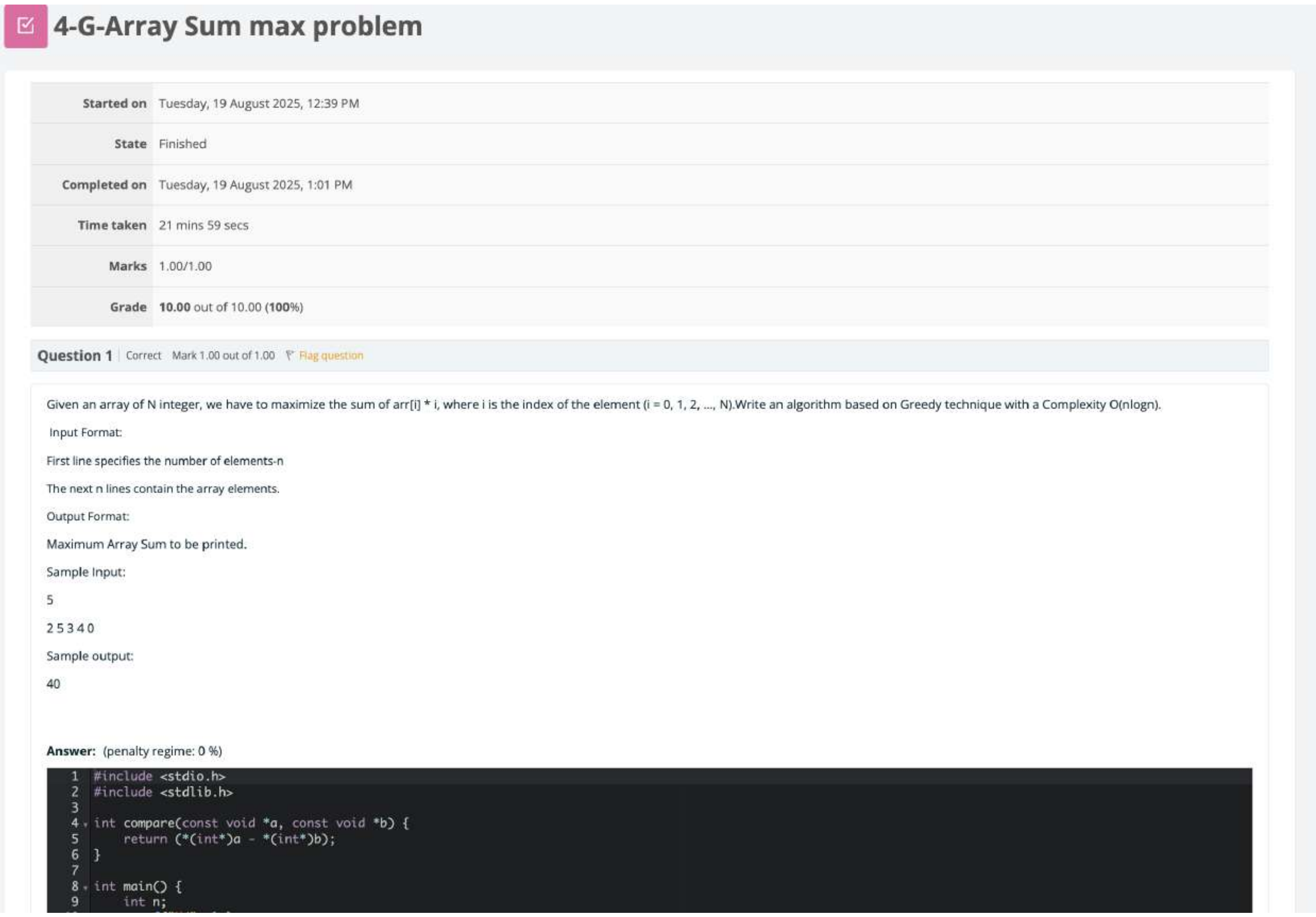
```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int cmp(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
}

int main() {
    int n;
    scanf("%d", &n);
    int calories[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &calories[i]);
    }
    qsort(calories, n, sizeof(int), cmp);
    long long min_distance = 0;
    for (int i = 0; i < n; i++) {
        min_distance += (long long)calories[i] * pow(n , i);
    }
    printf("%lld\n", min_distance);
    return 0;
}
```

| | Test | Input | Expected | Got | |
|---|---|---|---|---|---|
| ✓ | Test Case 1 | 3<br>1 3 2 | 18 | 18 | ✓ |
| ✓ | Test Case 2 | 4<br>7 4 9 6 | 389 | 389 | ✓ |
| ✓ | Test Case 3 | 3<br>5 10 7 | 76 | 76 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



# 4-G-Array Sum max problem

| | |
|---|---|
| **Started on** | Tuesday, 19 August 2025, 12:39 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 19 August 2025, 1:01 PM |
| **Time taken** | 21 mins 59 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 of N integer, we have to maximize the sum of arr[i] * i, where i is the index of the element (i = 0, 1, 2, ..., N).Write an algorithm based on Greedy technique with a Complexity O(nlogn).

Input Format:

First line specifies the number of elements-n

The next n lines contain the array elements.

Output Format:

Maximum Array Sum to be printed.

Sample Input:

5

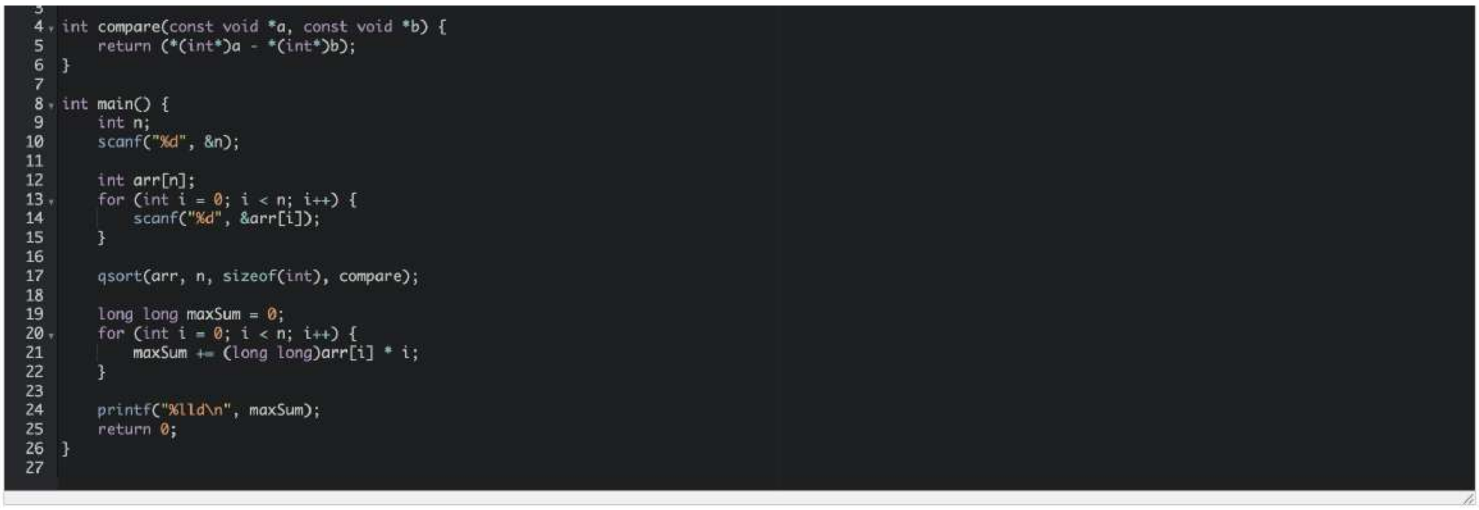
2 5 3 4 0

Sample output:

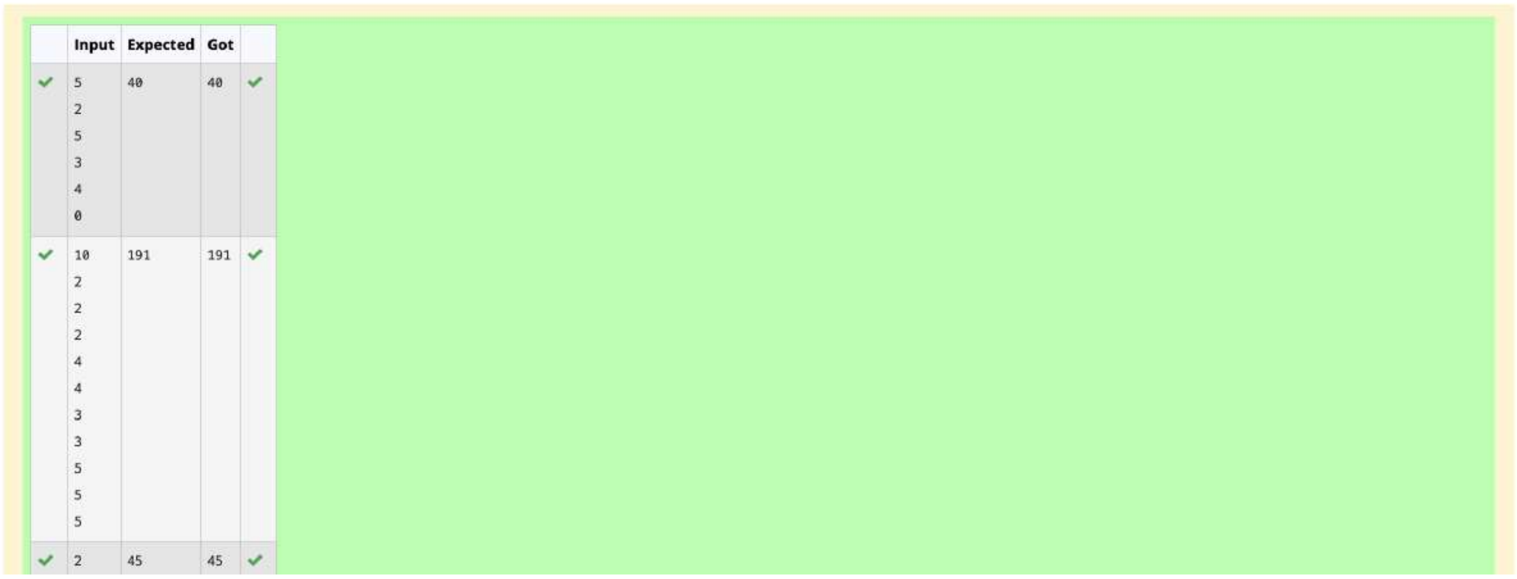
40

**Answer:** (penalty regime: 0 %)

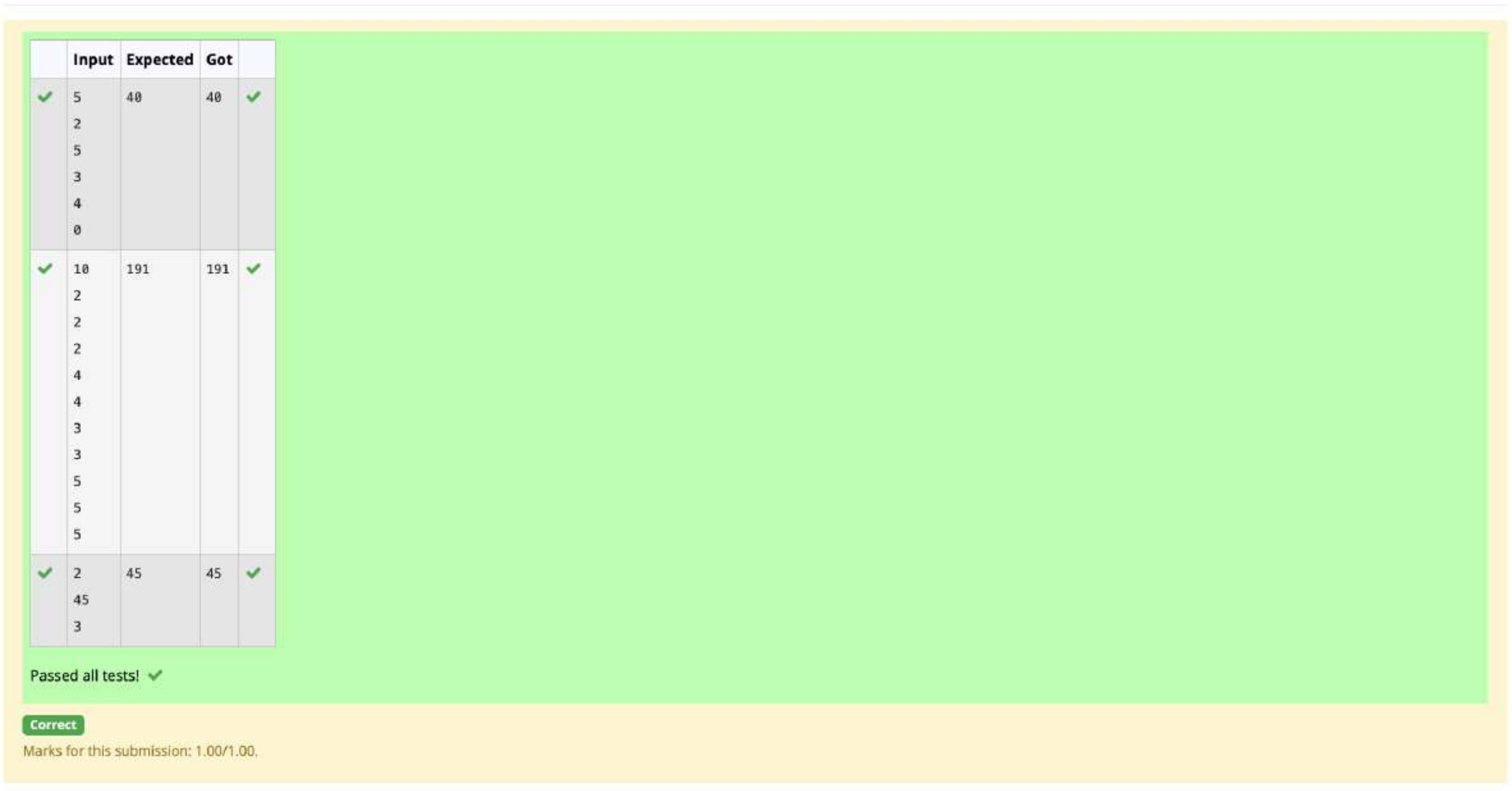
```c
#include <stdio.h>
#include <stdlib.h>

int compare(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}

int main() {
    int n;
```



```c
int compare(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}

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

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

    qsort(arr, n, sizeof(int), compare);

    long long maxSum = 0;
    for (int i = 0; i < n; i++) {
        maxSum += (long long)arr[i] * i;
    }

    printf("%lld\n", maxSum);
    return 0;
}
```



|  | Input | Expected | Got |  |
|---|---|---|---|---|
| ✓ | 5 2 5 3 4 0 | 40 | 40 | ✓ |
| ✓ | 10 2 2 2 4 4 3 3 5 5 5 | 191 | 191 | ✓ |
| ✓ | 2 | 45 | 45 | ✓ |



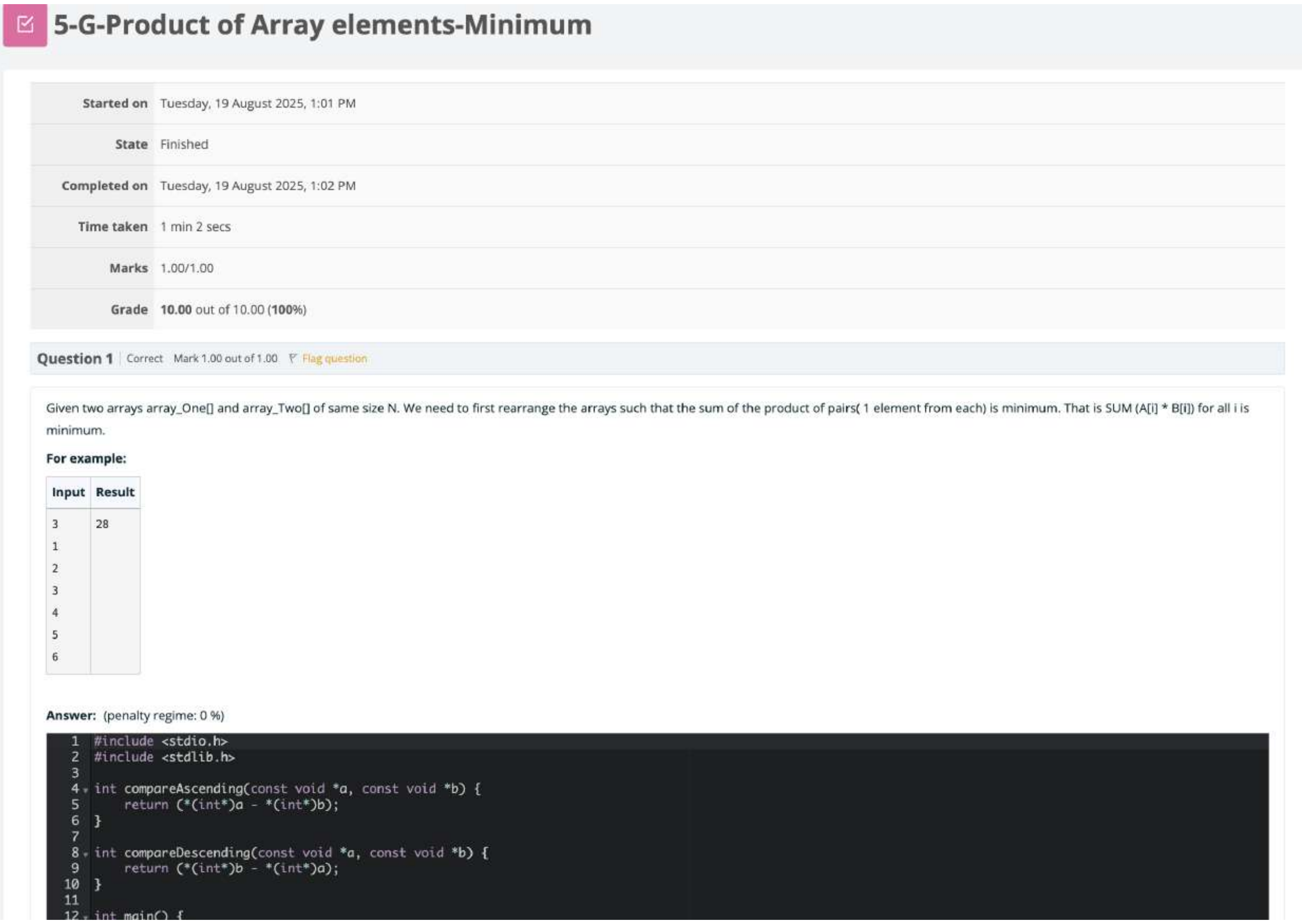
| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 5<br>2<br>5<br>3<br>4<br>0 | 40 | 40 | ✓ |
| ✓ | 10<br>2<br>2<br>2<br>4<br>4<br>3<br>3<br>5<br>5<br>5 | 191 | 191 | ✓ |
| ✓ | 2<br>45<br>3 | 45 | 45 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review



# 5-G-Product of Array elements-Minimum

| | |
|---|---|
| **Started on** | Tuesday, 19 August 2025, 1:01 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 19 August 2025, 1:02 PM |
| **Time taken** | 1 min 2 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

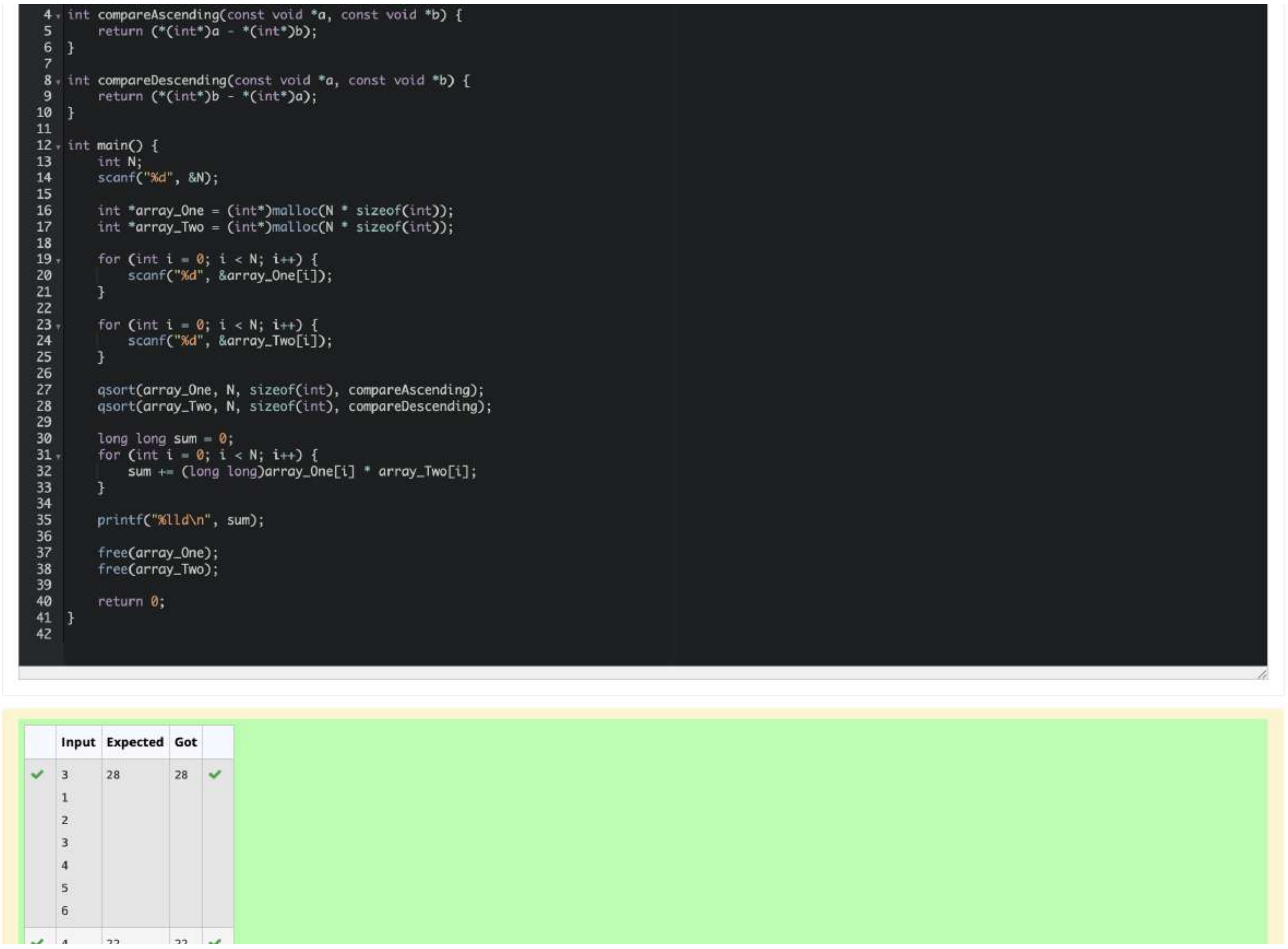
Given two arrays array_One[] and array_Two[] of same size N. We need to first rearrange the arrays such that the sum of the product of pairs( 1 element from each) is minimum. That is SUM (A[i] * B[i]) for all i is minimum.

**For example:**

| Input | Result |
|---|---|
| 3<br>1<br>2<br>3<br>4<br>5<br>6 | 28 |

**Answer:** (penalty regime: 0 %)

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

int compareAscending(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}

int compareDescending(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
}

int main() {
```



```c
int compareAscending(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}

int compareDescending(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
}

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

    int *array_One = (int*)malloc(N * sizeof(int));
    int *array_Two = (int*)malloc(N * sizeof(int));

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

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

    qsort(array_One, N, sizeof(int), compareAscending);
    qsort(array_Two, N, sizeof(int), compareDescending);

    long long sum = 0;
    for (int i = 0; i < N; i++) {
        sum += (long long)array_One[i] * array_Two[i];
    }

    printf("%lld\n", sum);

    free(array_One);
    free(array_Two);

    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 3<br>1<br>2<br>3<br>4<br>5<br>6 | 28 | 28 | ✓ |
| ✓ | 4 | 22 | 22 | ✓ |

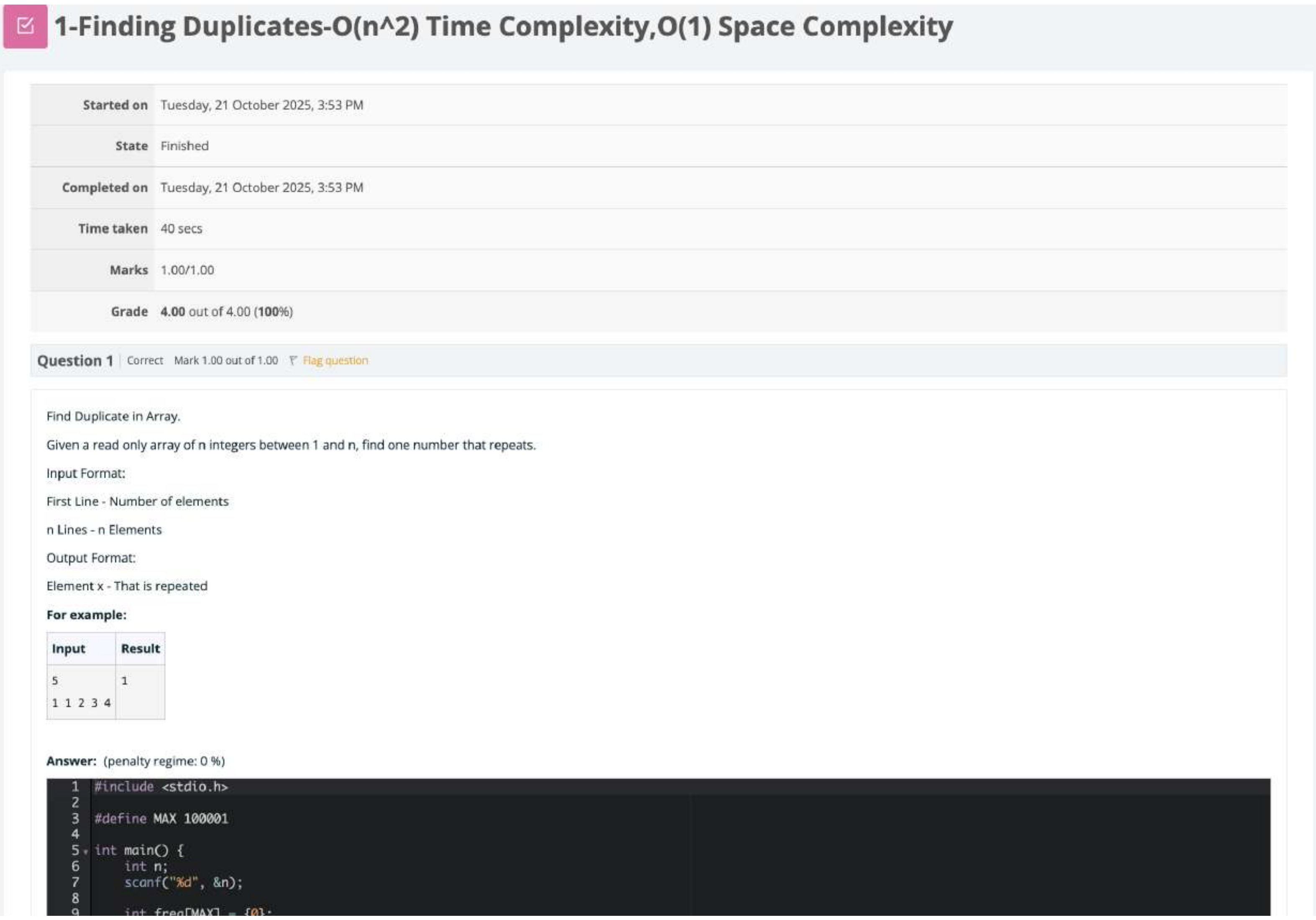
|  | Input | Expected | Got |  |
|---|---|---|---|---|
| ✓ | 3<br>1<br>2<br>3<br>4<br>5<br>6 | 28 | 28 | ✓ |
| ✓ | 4<br>7<br>5<br>1<br>2<br>1<br>3<br>4<br>1 | 22 | 22 | ✓ |
| ✓ | 5<br>20<br>10<br>30<br>10<br>40<br>8<br>9<br>4<br>3<br>10 | 590 | 590 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review



# 1-Finding Duplicates-O(n^2) Time Complexity,O(1) Space Complexity

| | |
|---|---|
| **Started on** | Tuesday, 21 October 2025, 3:53 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 21 October 2025, 3:53 PM |
| **Time taken** | 40 secs |
| **Marks** | 1.00/1.00 |
| **Grade** | **4.00** out of 4.00 (**100**%) |

**Question 1** Correct Mark 1.00 out of 1.00 Flag question

Find Duplicate in Array.

Given a read only array of n integers between 1 and n, find one number that repeats.

Input Format:

First Line - Number of elements

n Lines - n Elements

Output Format:

Element x - That is repeated

**For example:**

| Input | Result |
|---|---|
| 5<br>1 1 2 3 4 | 1 |

**Answer:** (penalty regime: 0 %)

```
#include <stdio.h>

#define MAX 100001

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

    int freq[MAX] = {0};
```

| 1 1 2 3 4 | |
|---|---|

**Answer:** (penalty regime: 0 %)

```c
#include <stdio.h>

#define MAX 100001

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

    int freq[MAX] = {0};
    int x;

    for (int i = 0; i < n; i++) {
        scanf("%d", &x);
        if (freq[x] == 1) {
            printf("%d\n", x);
            return 0;
        }
        freq[x]++;
    }

    return 0;
}
```

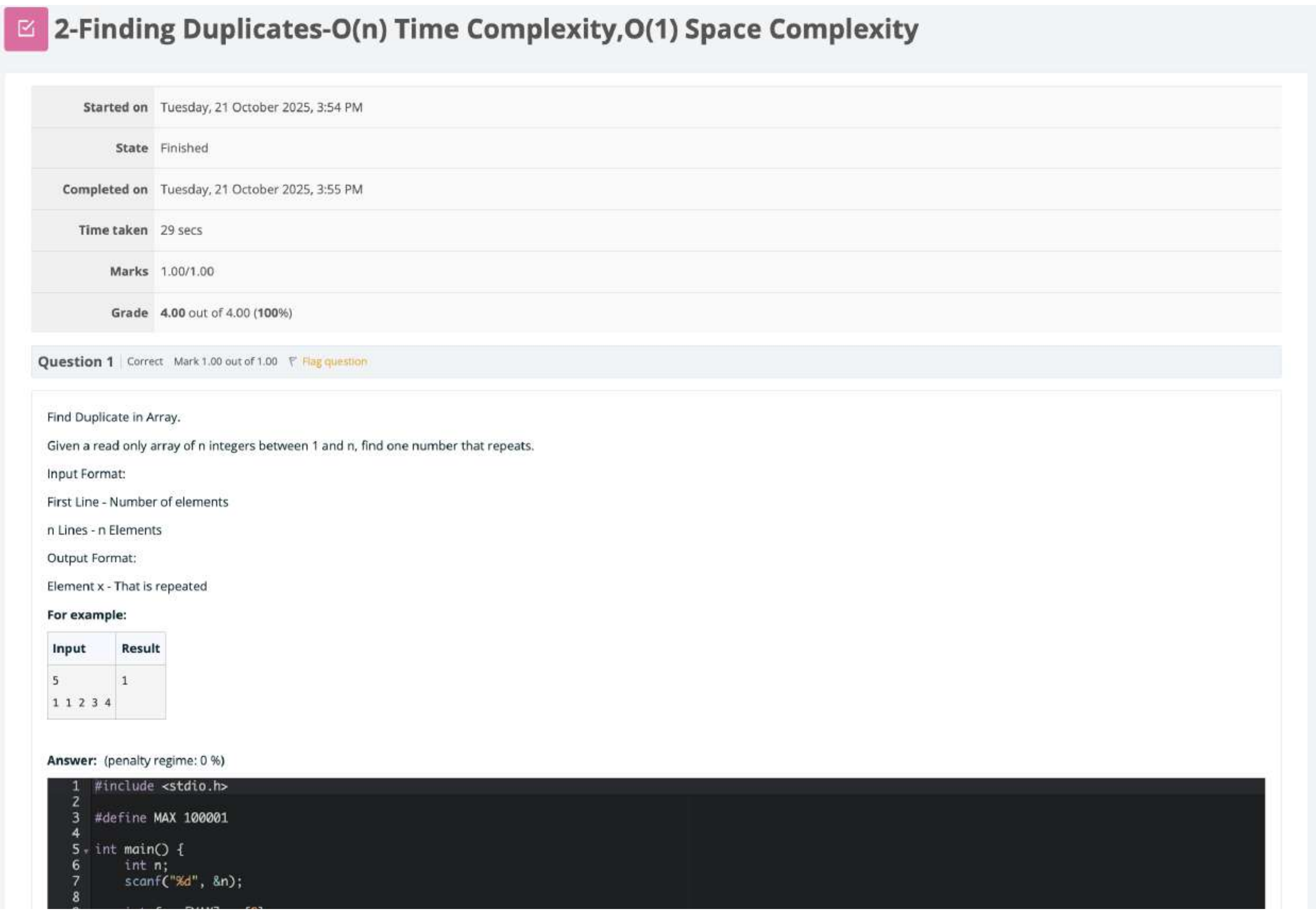
| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 11<br>10 9 7 6 5 1 2 3 8 4 7 | 7 | 7 | ✓ |
| ✓ | 5<br>1 2 3 4 4 | 4 | 4 | ✓ |
| ✓ | 5<br>1 1 2 3 4 | 1 | 1 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review



# 2-Finding Duplicates-O(n) Time Complexity,O(1) Space Complexity

| | |
|---|---|
| **Started on** | Tuesday, 21 October 2025, 3:54 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 21 October 2025, 3:55 PM |
| **Time taken** | 29 secs |
| **Marks** | 1.00/1.00 |
| **Grade** | **4.00** out of 4.00 (**100**%) |

**Question 1** Correct Mark 1.00 out of 1.00 Flag question

Find Duplicate in Array.

Given a read only array of n integers between 1 and n, find one number that repeats.

Input Format:

First Line - Number of elements

n Lines - n Elements

Output Format:

Element x - That is repeated

**For example:**

| Input | Result |
|---|---|
| 5<br>1 1 2 3 4 | 1 |

**Answer:** (penalty regime: 0 %)

```
#include <stdio.h>

#define MAX 100001

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

```

**Answer:** (penalty regime: 0 %)

```c
#include <stdio.h>

#define MAX 100001

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

    int freq[MAX] = {0};
    int x;

    for (int i = 0; i < n; i++) {
        scanf("%d", &x);
        if (freq[x] == 1) {
            printf("%d\n", x);
            return 0;
        }
        freq[x]++;
    }

    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 11<br>10 9 7 6 5 1 2 3 8 4 7 | 7 | 7 | ✓ |
| ✓ | 5<br>1 2 3 4 4 | 4 | 4 | ✓ |
| ✓ | 5<br>1 1 2 3 4 | 1 | 1 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review

# 3-Print Intersection of 2 sorted arrays-O(m*n)Time Complexity,O(1) Space Complexity

| | |
|---|---|
| **Started on** | Tuesday, 21 October 2025, 3:55 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 21 October 2025, 3:56 PM |
| **Time taken** | 1 min 2 secs |
| **Marks** | 1.00/1.00 |
| **Grade** | **30.00** out of 30.00 (**100**%) |

**Question 1** Correct Mark 1.00 out of 1.00 Flag question

Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

- The first line contains T, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

6 1 2 3 4 5 6

2 1 6

Output:

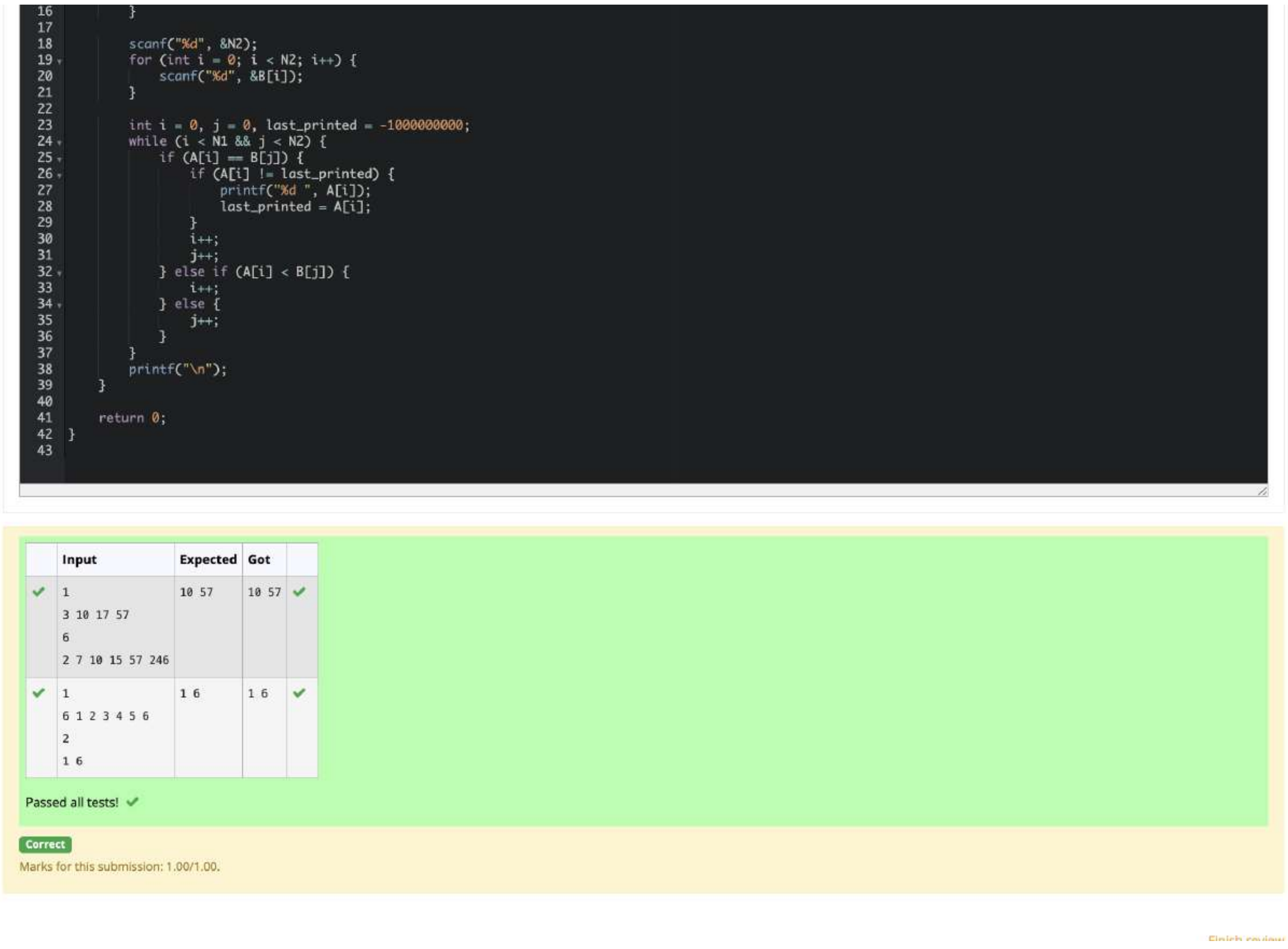
1 6

**For example:**

| Input | Result |
|---|---|
| 1<br>3 10 17 57<br>6<br>2 7 10 15 57 246 | 10 57 |

**Answer:** (penalty regime: 0 %)

```c
#include <stdio.h>

#define MAX 10000

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

    while (T--) {
        int N1, N2;
        int A[MAX], B[MAX];

        scanf("%d", &N1);
        for (int i = 0; i < N1; i++) {
            scanf("%d", &A[i]);
        }

        scanf("%d", &N2);
        for (int i = 0; i < N2; i++) {
            scanf("%d", &B[i]);
        }

        int i = 0, j = 0, last_printed = -1000000000;
        while (i < N1 && j < N2) {
            if (A[i] == B[j]) {
                if (A[i] != last_printed) {
                    printf("%d ", A[i]);
                    last_printed = A[i];
                }
                i++;
                j++;
            } else if (A[i] < B[j]) {
                i++;
```



```c
        }

        scanf("%d", &N2);
        for (int i = 0; i < N2; i++) {
            scanf("%d", &B[i]);
        }

        int i = 0, j = 0, last_printed = -1000000000;
        while (i < N1 && j < N2) {
            if (A[i] == B[j]) {
                if (A[i] != last_printed) {
                    printf("%d ", A[i]);
                    last_printed = A[i];
                }
                i++;
                j++;
            } else if (A[i] < B[j]) {
                i++;
            } else {
                j++;
            }
        }
        printf("\n");
    }

    return 0;
}

```

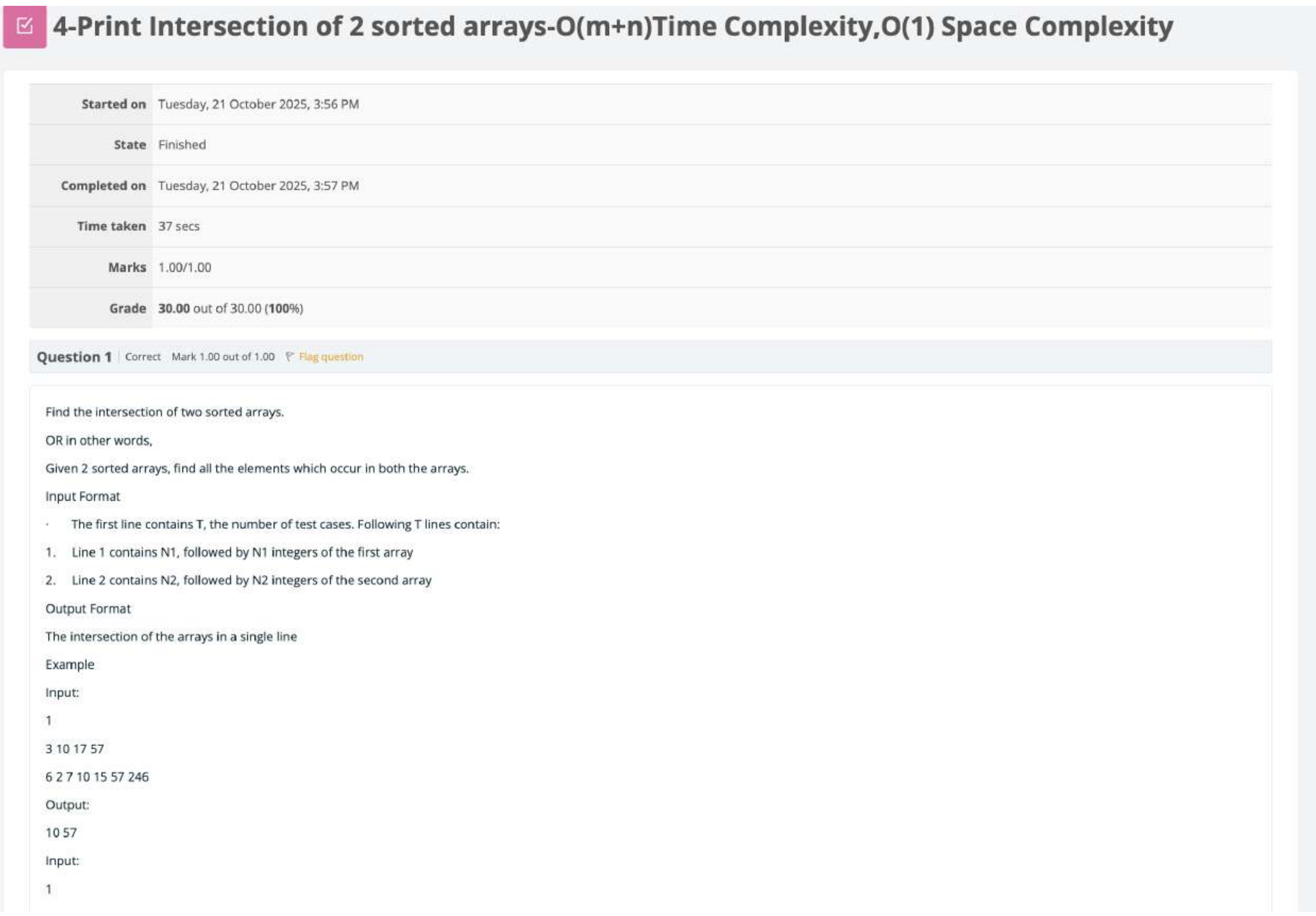
| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 1 3 10 17 57 6 2 7 10 15 57 246 | 10 57 | 10 57 | ✓ |
| ✓ | 1 6 1 2 3 4 5 6 2 1 6 | 1 6 | 1 6 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review



# 4-Print Intersection of 2 sorted arrays-O(m+n)Time Complexity,O(1) Space Complexity

| | |
|---|---|
| **Started on** | Tuesday, 21 October 2025, 3:56 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 21 October 2025, 3:57 PM |
| **Time taken** | 37 secs |
| **Marks** | 1.00/1.00 |
| **Grade** | **30.00** out of 30.00 (**100**%) |

**Question 1** Correct Mark 1.00 out of 1.00 Flag question

Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

- The first line contains **T**, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

6 1 2 3 4 5 6

2 1 6

Output:

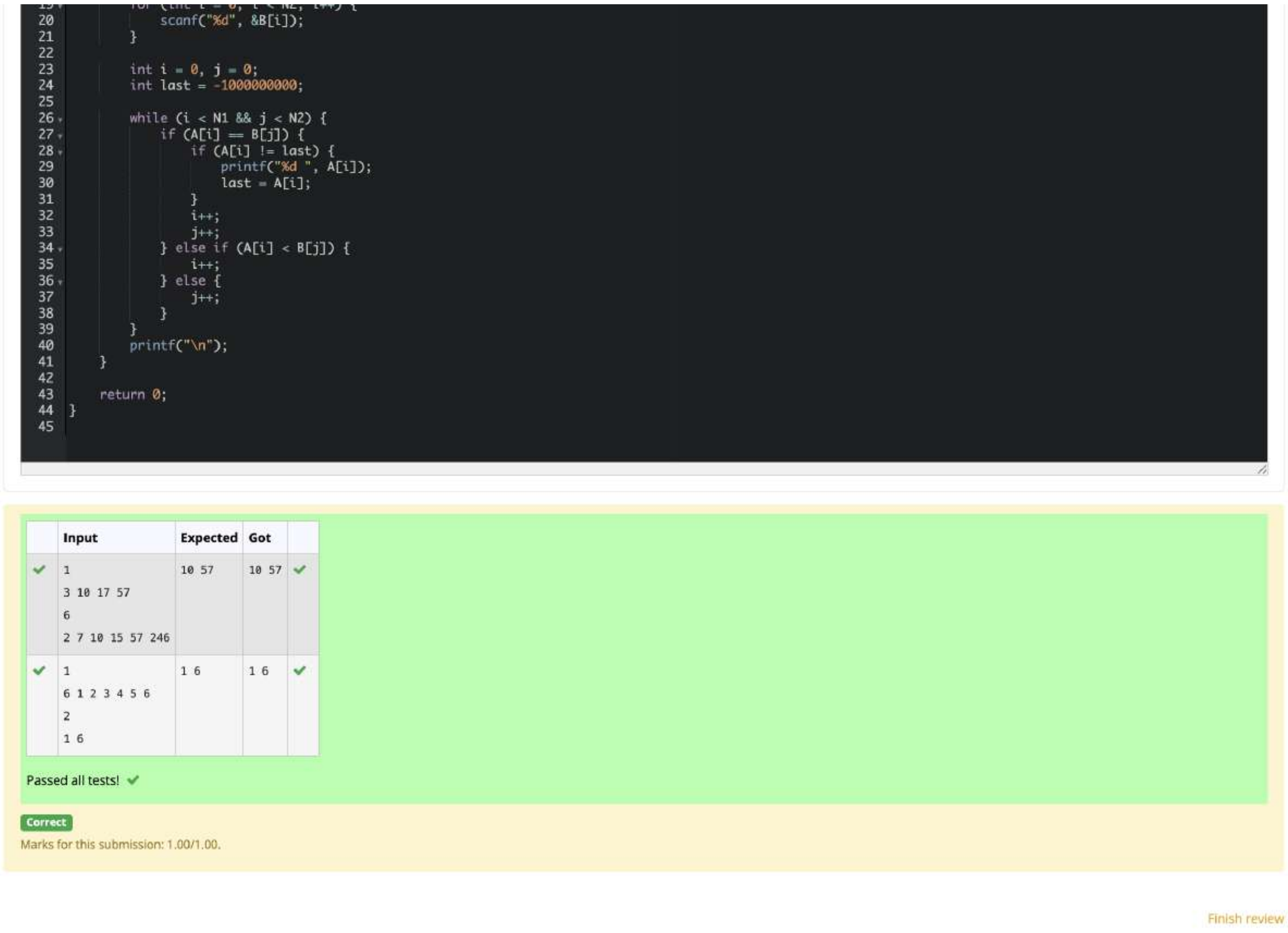
1 6

**For example:**

| Input | Result |
|---|---|
| 1<br>3 10 17 57<br>6<br>2 7 10 15 57 246 | 10 57 |

**Answer:** (penalty regime: 0 %)

```c
#include <stdio.h>

#define MAX 10000

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

    while (T--) {
        int N1, N2;
        int A[MAX], B[MAX];

        scanf("%d", &N1);
        for (int i = 0; i < N1; i++) {
            scanf("%d", &A[i]);
        }

        scanf("%d", &N2);
        for (int i = 0; i < N2; i++) {
            scanf("%d", &B[i]);
        }

        int i = 0, j = 0;
        int last = -1000000000;

        while (i < N1 && j < N2) {
            if (A[i] == B[j]) {
                if (A[i] != last) {
                    printf("%d ", A[i]);
                    last = A[i];
                }
                i++;
                j++;
            } else if (A[i] < B[j]) {
```



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

        int i = 0, j = 0;
        int last = -1000000000;

        while (i < N1 && j < N2) {
            if (A[i] == B[j]) {
                if (A[i] != last) {
                    printf("%d ", A[i]);
                    last = A[i];
                }
                i++;
                j++;
            } else if (A[i] < B[j]) {
                i++;
            } else {
                j++;
            }
        }
        printf("\n");
    }

    return 0;
}
```

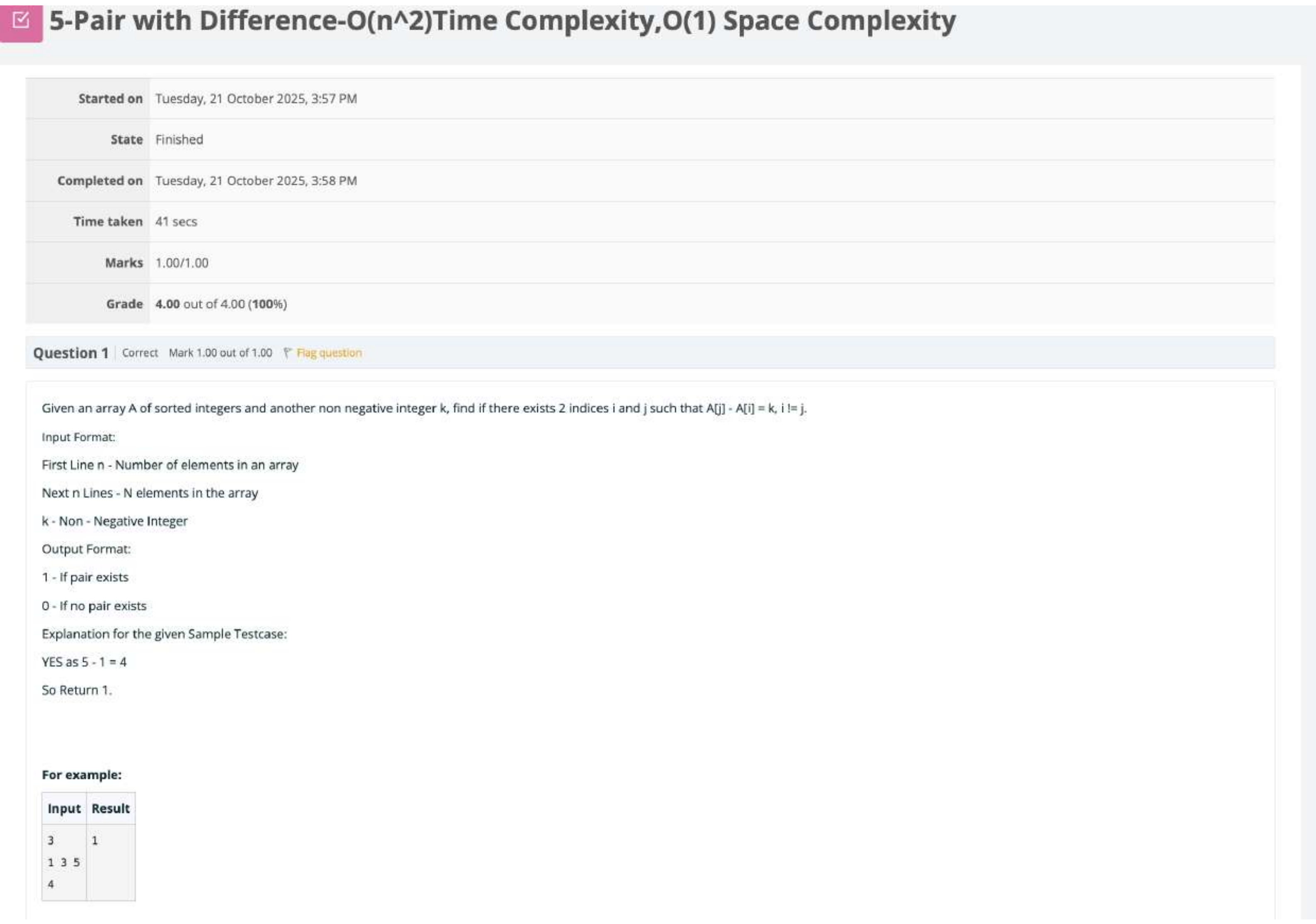
| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 1<br>3 10 17 57<br>6<br>2 7 10 15 57 246 | 10 57 | 10 57 | ✓ |
| ✓ | 1<br>6 1 2 3 4 5 6<br>2<br>1 6 | 1 6 | 1 6 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review



# 5-Pair with Difference-O(n^2)Time Complexity,O(1) Space Complexity

| | |
|---|---|
| **Started on** | Tuesday, 21 October 2025, 3:57 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 21 October 2025, 3:58 PM |
| **Time taken** | 41 secs |
| **Marks** | 1.00/1.00 |
| **Grade** | **4.00** out of 4.00 (**100%**) |

**Question 1** Correct Mark 1.00 out of 1.00 Flag question

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[j] - A[i] = k, i != j.

Input Format:

First Line n - Number of elements in an array

Next n Lines - N elements in the array

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

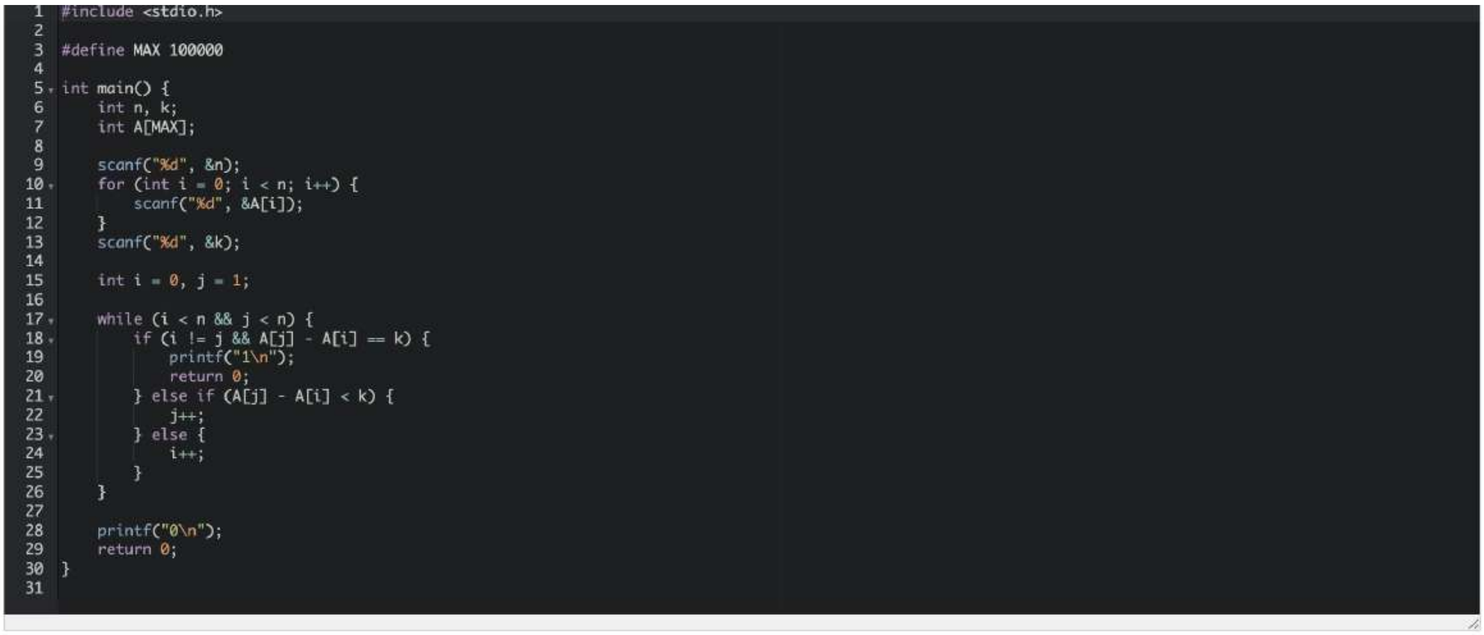
Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

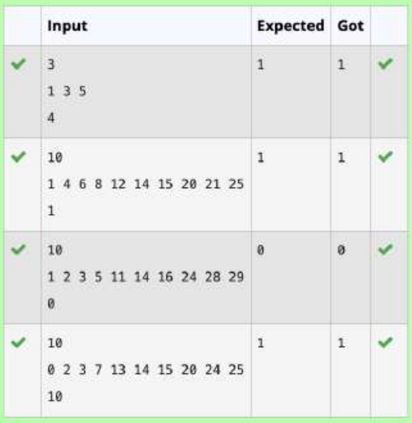
So Return 1.

**For example:**

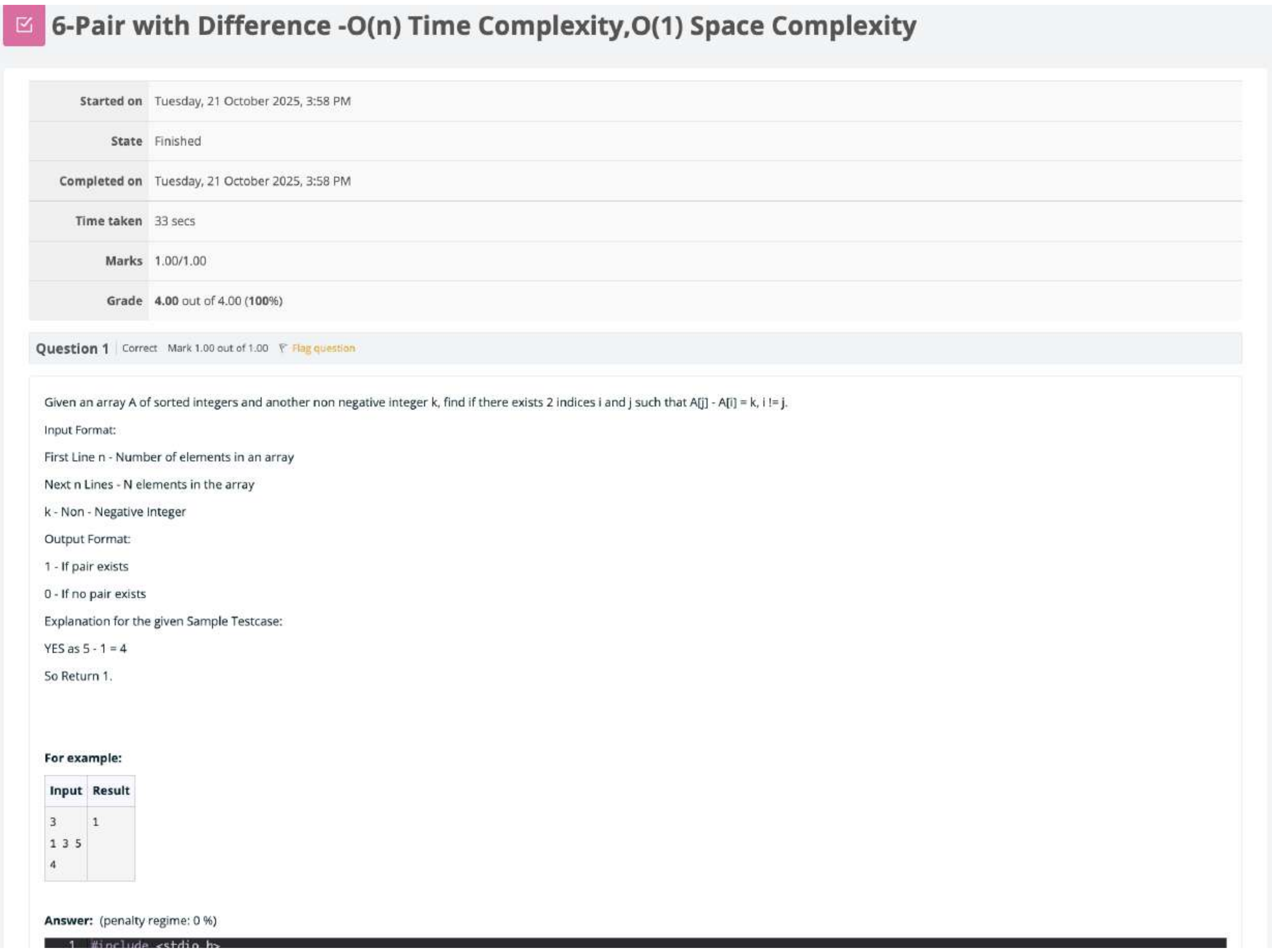
| Input | Result |
|---|---|
| 3<br>1 3 5<br>4 | 1 |



```c
#include <stdio.h>

#define MAX 100000

int main() {
    int n, k;
    int A[MAX];

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

    int i = 0, j = 1;

    while (i < n && j < n) {
        if (i != j && A[j] - A[i] == k) {
            printf("1\n");
            return 0;
        } else if (A[j] - A[i] < k) {
            j++;
        } else {
            i++;
        }
    }

    printf("0\n");
    return 0;
}

```



| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 3<br>1 3 5<br>4 | 1 | 1 | ✓ |
| ✓ | 10<br>1 4 6 8 12 14 15 20 21 25<br>1 | 1 | 1 | ✓ |
| ✓ | 10<br>1 2 3 5 11 14 16 24 28 29<br>0 | 0 | 0 | ✓ |
| ✓ | 10<br>0 2 3 7 13 14 15 20 24 25<br>10 | 1 | 1 | ✓ |



# 6-Pair with Difference -O(n) Time Complexity,O(1) Space Complexity

| | |
|---|---|
| **Started on** | Tuesday, 21 October 2025, 3:58 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 21 October 2025, 3:58 PM |
| **Time taken** | 33 secs |
| **Marks** | 1.00/1.00 |
| **Grade** | **4.00** out of 4.00 (**100**%) |

**Question 1** Correct Mark 1.00 out of 1.00 Flag question

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[j] - A[i] = k, i != j.

Input Format:

First Line n - Number of elements in an array

Next n Lines - N elements in the array

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

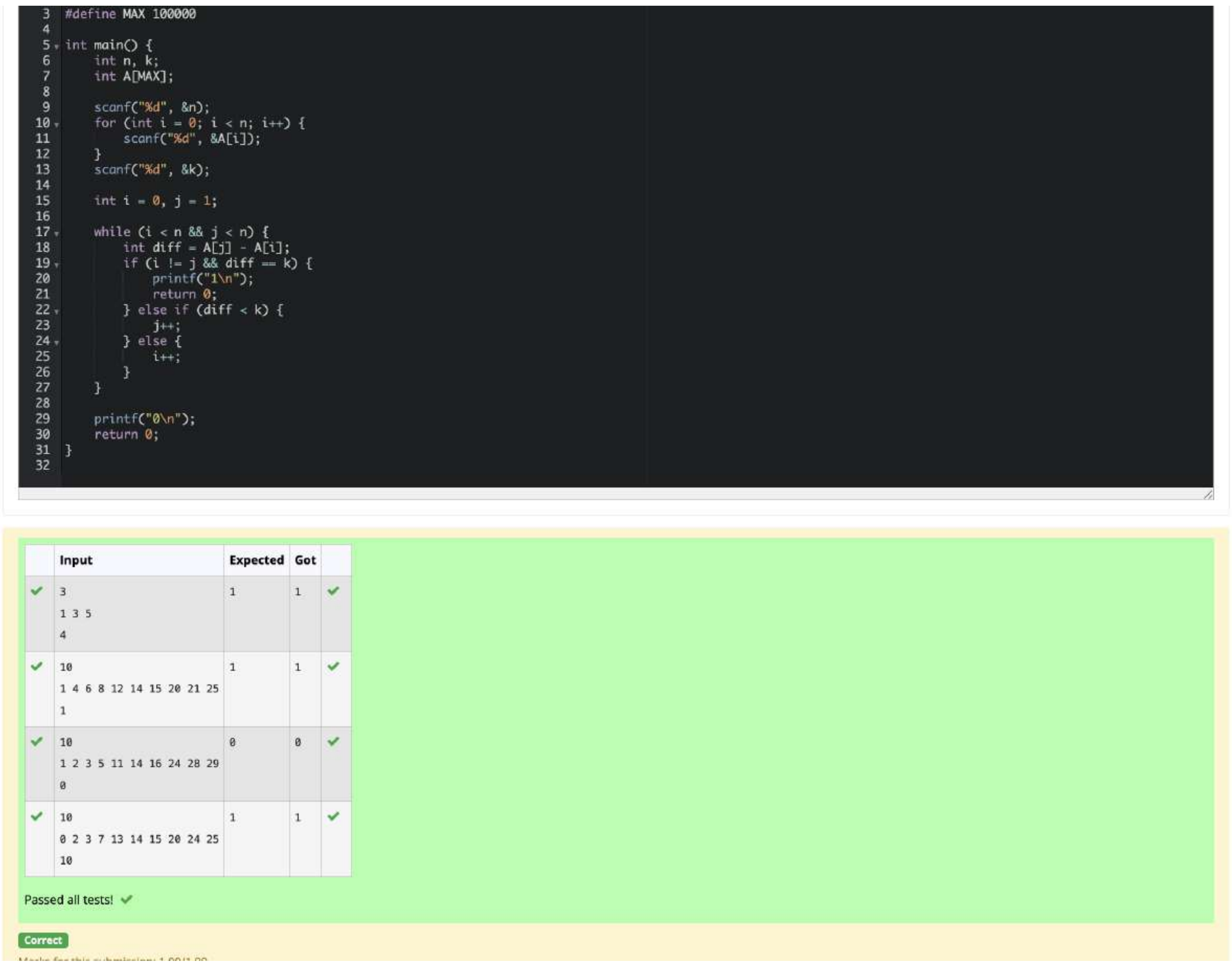
So Return 1.

**For example:**

| Input | Result |
|---|---|
| 3<br>1 3 5<br>4 | 1 |

**Answer:** (penalty regime: 0 %)

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



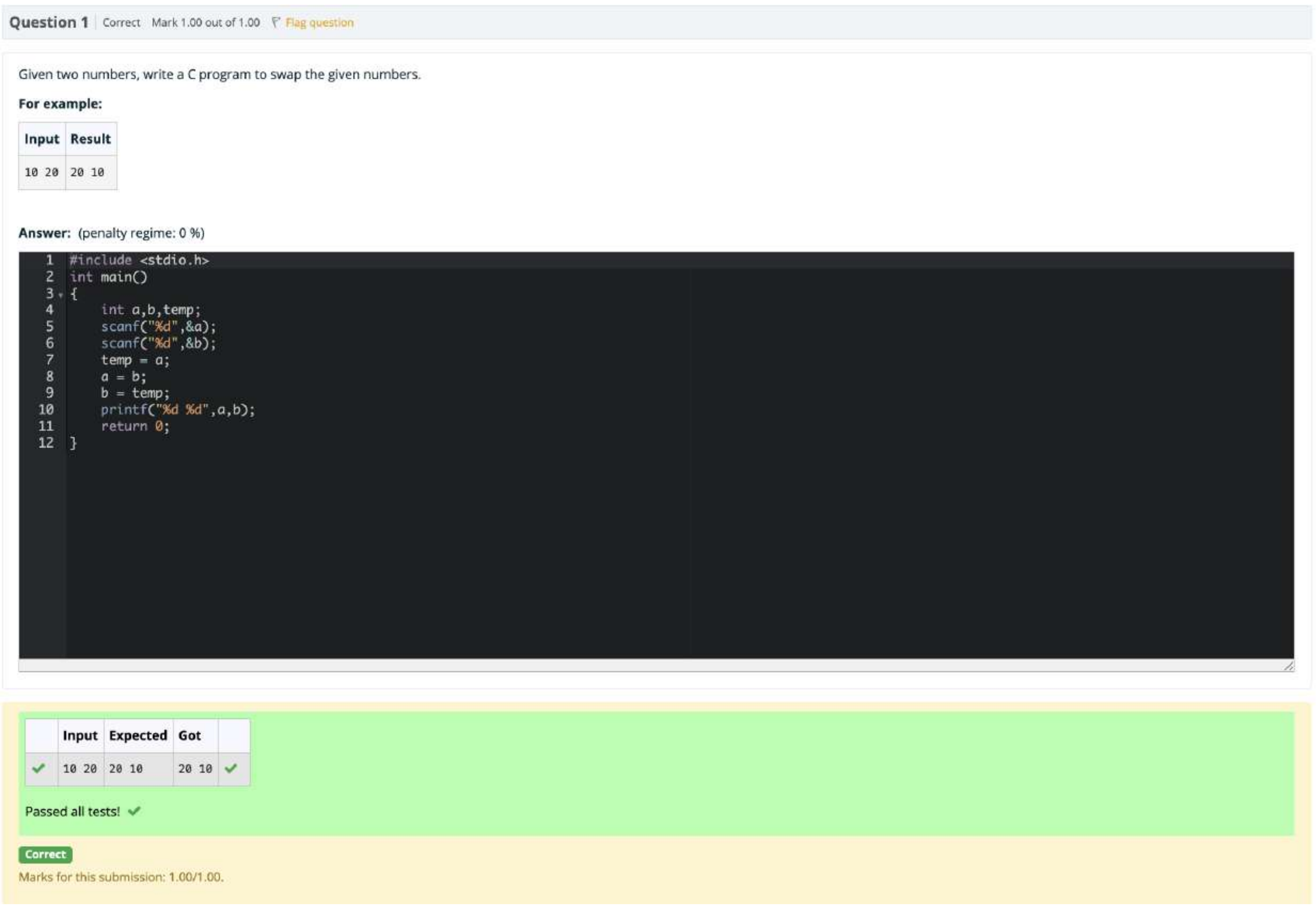
```c
#define MAX 100000

int main() {
    int n, k;
    int A[MAX];

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

    int i = 0, j = 1;

    while (i < n && j < n) {
        int diff = A[j] - A[i];
        if (i != j && diff == k) {
            printf("1\n");
            return 0;
        } else if (diff < k) {
            j++;
        } else {
            i++;
        }
    }

    printf("0\n");
    return 0;
}

```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 3<br>1 3 5<br>4 | 1 | 1 | ✓ |
| ✓ | 10<br>1 4 6 8 12 14 15 20 21 25<br>1 | 1 | 1 | ✓ |
| ✓ | 10<br>1 2 3 5 11 14 16 24 28 29<br>0 | 0 | 0 | ✓ |
| ✓ | 10<br>0 2 3 7 13 14 15 20 24 25<br>10 | 1 | 1 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00


**Question 1** Correct Mark 1.00 out of 1.00 Flag question

Given two numbers, write a C program to swap the given numbers.

**For example:**

| Input | Result |
|---|---|
| 10 20 | 20 10 |

**Answer:** (penalty regime: 0 %)

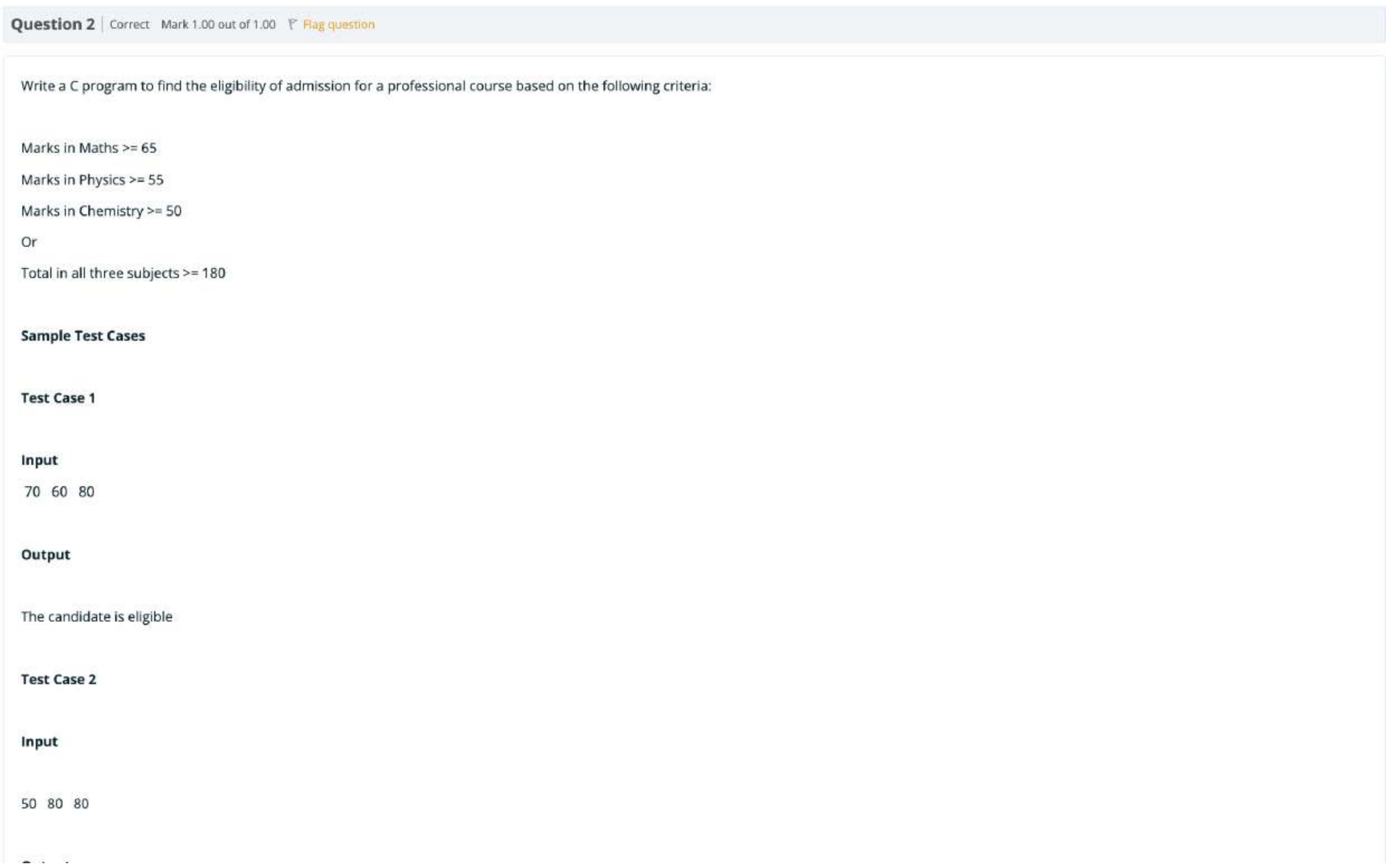
```c
#include <stdio.h>
int main()
{
    int a,b,temp;
    scanf("%d",&a);
    scanf("%d",&b);
    temp = a;
    a = b;
    b = temp;
    printf("%d %d",a,b);
    return 0;
}
```

|  | Input | Expected | Got |  |
|---|---|---|---|---|
| ✓ | 10 20 | 20 10 | 20 10 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



**Question 2** | Correct Mark 1.00 out of 1.00 Flag question

Write a C program to find the eligibility of admission for a professional course based on the following criteria:

Marks in Maths >= 65

Marks in Physics >= 55

Marks in Chemistry >= 50

Or

Total in all three subjects >= 180

**Sample Test Cases**

**Test Case 1**

**Input**

```
 70  60  80
```

**Output**

```
The candidate is eligible
```

**Test Case 2**

**Input**

```
50  80  80
```

**Output**

The candidate is eligible

**Test Case 3**

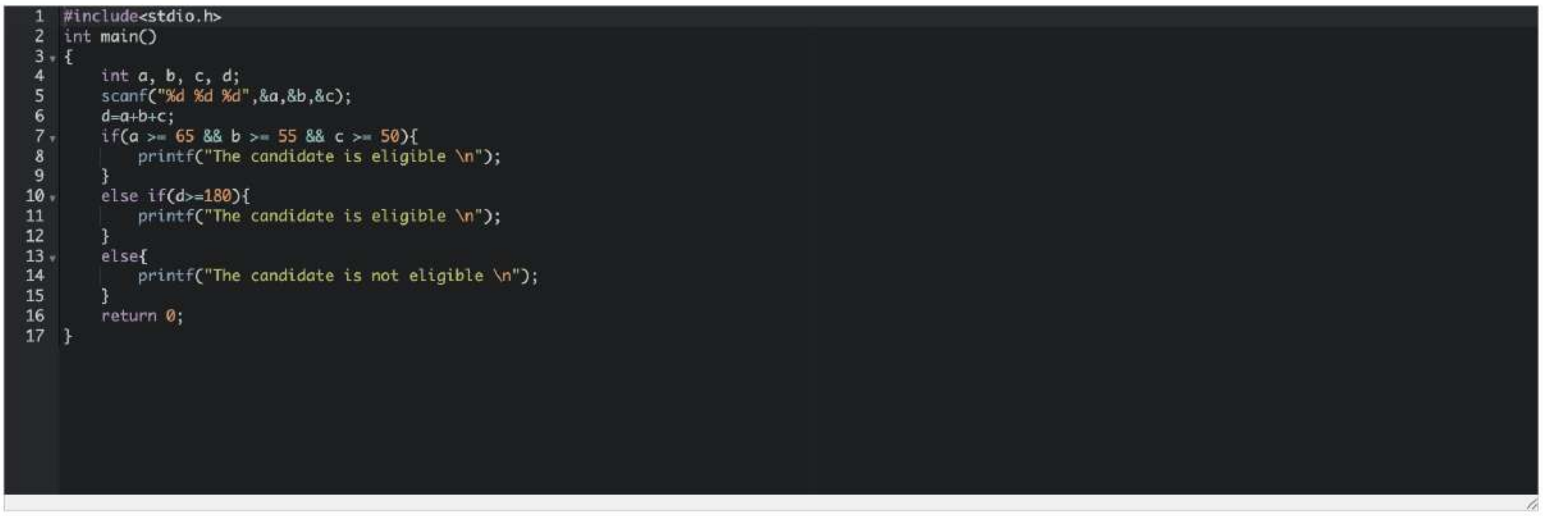
**Input**

50 60 40

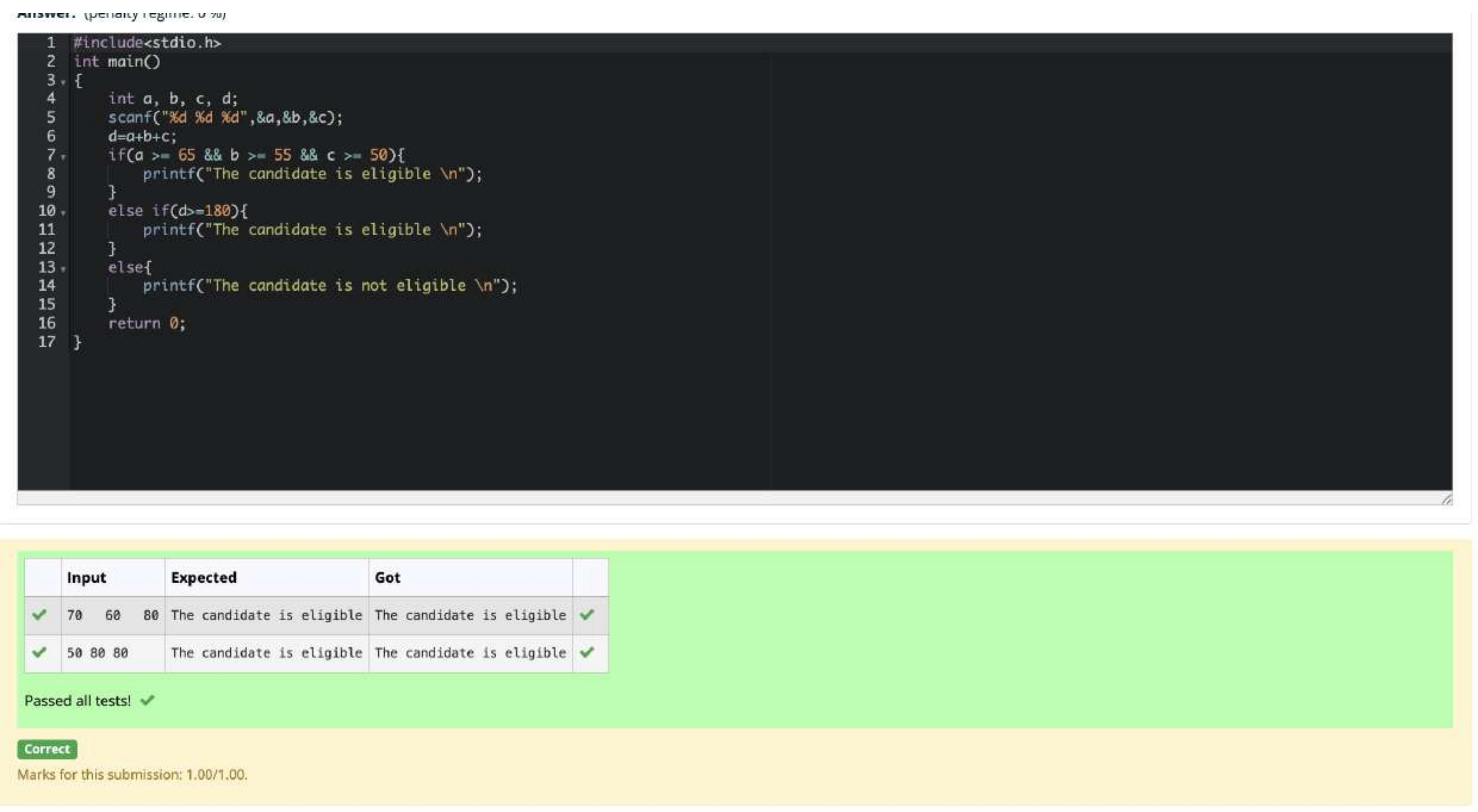
**Output**

The candidate is not eligible

**Answer:** (penalty regime: 0 %)



```
#include<stdio.h>
int main()
{
    int a, b, c, d;
    scanf("%d %d %d",&a,&b,&c);
    d=a+b+c;
    if(a >= 65 && b >= 55 && c >= 50){
        printf("The candidate is eligible \n");
    }
    else if(d>=180){
        printf("The candidate is eligible \n");
    }
    else{
        printf("The candidate is not eligible \n");
    }
    return 0;
}
```



**Answer:** (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int a, b, c, d;
    scanf("%d %d %d",&a,&b,&c);
    d=a+b+c;
    if(a >= 65 && b >= 55 && c >= 50){
        printf("The candidate is eligible \n");
    }
    else if(d>=180){
        printf("The candidate is eligible \n");
    }
    else{
        printf("The candidate is not eligible \n");
    }
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 70    60    80 | The candidate is eligible | The candidate is eligible | ✓ |
| ✓ | 50 80 80 | The candidate is eligible | The candidate is eligible | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

**Question 3** Correct Mark 1.00 out of 1.00 Flag question

Malini goes to BestSave hyper market to buy grocery items. BestSave hyper market provides 10% discount on the bill amount B when ever the bill amount B is more than Rs.2000.

The bill amount B is passed as the input to the program. The program must print the final amount A payable by Malini.

Input Format:

The first line denotes the value of B.

Output Format:

The first line contains the value of the final payable amount A.

Example Input/Output 1:

Input:

1900

Output:

1900

Example Input/Output 2:

Input:

3000

Output:

3000

Output:

2700

**Answer:** (penalty regime: 0 %)

```c
#include<stdio.h>
int main()
{
    int a, b, c;
    scanf("%d",&a);
    b = a/10;
    c=a-b;
    if(a >2000 ){
        printf("%d",c);
    }
    else{
        printf("%d",a);
    }
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 1900 | 1900 | 1900 | ✓ |
| ✓ | 3000 | 2700 | 2700 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

**Question 4** Correct Mark 1.00 out of 1.00 Flag question

Baba is very kind to beggars and every day Baba donates half of the amount he has when ever a beggar requests him. The money M left in Baba's hand is passed as the input and the number of beggars B who received the alms are passed as the input. The program must print the money Baba had in the beginning of the day.

**Input Format:**

The first line denotes the value of M.
The second line denotes the value of B.

**Output Format:**

The first line denotes the value of money with Baba in the beginning of the day.

**Example Input/Output:**

Input:

```
100
2
```

Output:

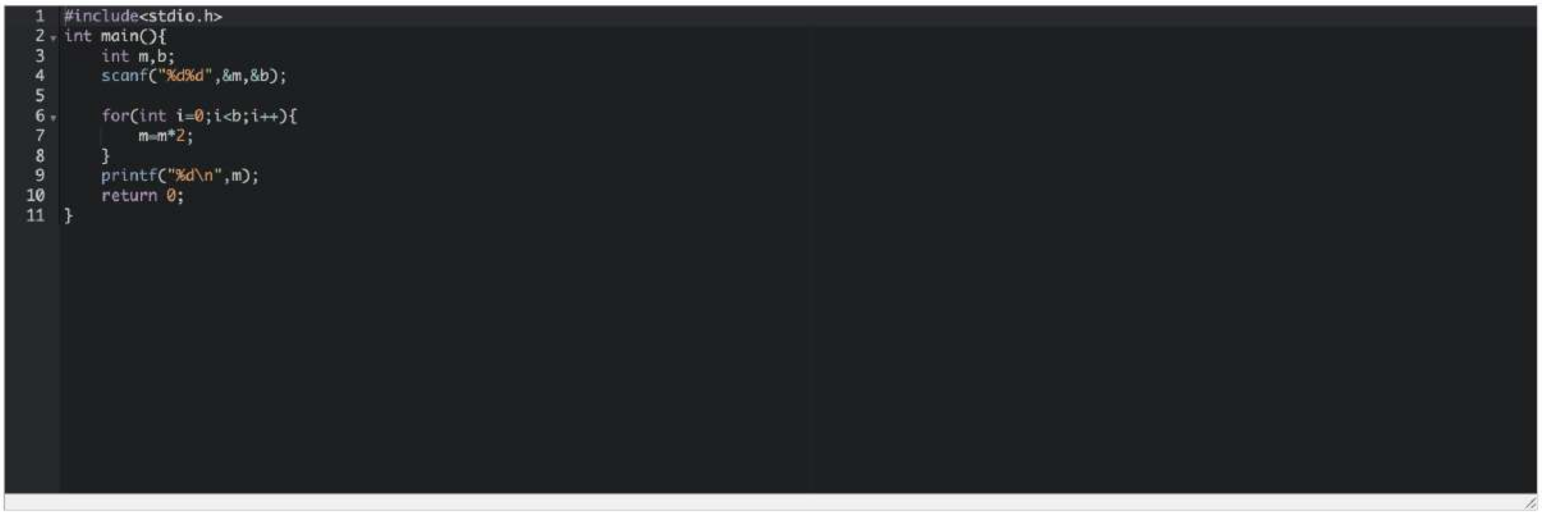
```
400
```

Explanation:

Baba donated to two beggars. So when he encountered second beggar he had 100*2 = Rs.200 and when he encountered 1st he had 200*2 = Rs.400.

**Answer:** (penalty regime: 0 %)

**Answer:** (penalty regime: 0 %)



```
#include<stdio.h>
int main(){
    int m,b;
    scanf("%d%d",&m,&b);

    for(int i=0;i<b;i++){
        m=m*2;
    }
    printf("%d\n",m);
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 100<br>2 | 400 | 400 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

**Question 5** | Correct Mark 1.00 out of 1.00 Flag question

The CEO of company ABC Inc wanted to encourage the employees coming on time to the office. So he announced that for every consecutive day an employee comes on time in a week (starting from Monday to Saturday), he will be awarded Rs.200 more than the previous day as "Punctuality Incentive". The incentive I for the starting day (ie on Monday) is passed as the input to the program. The number of days N an employee came on time consecutively starting from Monday is also passed as the input. The program must calculate and print the "Punctuality Incentive" P of the employee.

**Input Format:**

The first line denotes the value of I.
The second line denotes the value of N.

**Output Format:**

The first line denotes the value of P.

**Example Input/Output:**

Input:

```
500
3
```

Output:

```
2100
```

Explanation:

On Monday the employee receives Rs.500, on Tuesday Rs.700, on Wednesday Rs.900

On Monday the employee receives Rs.500, on Tuesday Rs.700, on Wednesday Rs.900

So total = Rs.2100

**Answer:** (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int x,n;
    scanf("%d%d",&x,&n);
    int total =0;
    for(int i=0;i < n;i++){
        total+=x+(i*200);
    }
    printf("%d\n",total);
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 500 3 | 2100 | 2100 | ✓ |
| ✓ | 100 3 | 900 | 900 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

# 1-DP-Playing with Numbers

| | |
|---|---|
| **Started on** | Friday, 31 October 2025, 1:29 AM |
| **State** | Finished |
| **Completed on** | Friday, 31 October 2025, 1:30 AM |
| **Time taken** | 13 secs |
| **Grade** | **10.00** out of 10.00 (**100%**) |

## Question 1

Correct Mark 10.00 out of 10.00 Flag question

**Playing with Numbers:**

Ram and Sita are playing with numbers by giving puzzles to each other. Now it was Ram term, so he gave Sita a positive integer 'n' and two numbers 1 and 3. He asked her to find the possible ways by which the number n can be represented using 1 and 3.Write any efficient algorithm to find the possible ways.

**Example 1:**

***Input:*** *6*
***Output:****6*
***Explanation:*** *There are 6 ways to 6 represent number with 1 and 3*

*1+1+1+1+1+1*
*3+3*
*1+1+1+3*
*1+1+3+1*
*1+3+1+1*
*3+1+1+1*

**Input Format**
First Line contains the number n

**Output Format**

**Print: The number of possible ways 'n' can be represented using 1 and 3**

Sample Input

6

Sample Output

6

Sample Output

6

**Answer:** (penalty regime: 0 %)

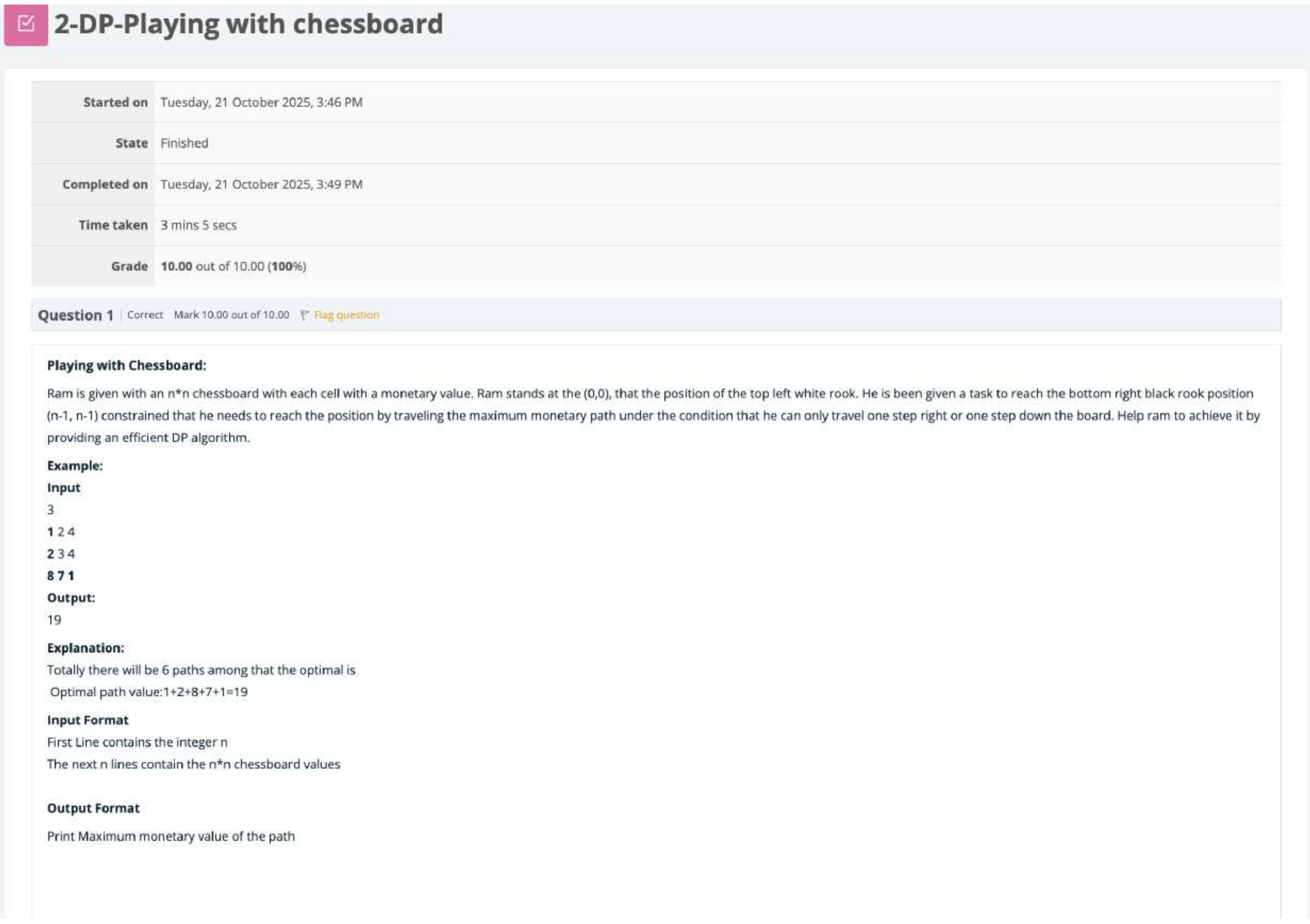
```c
#include <stdio.h>

unsigned long long countWays(int n) {
    unsigned long long dp[n + 1];
    dp[0] = 1;

    for (int i = 1; i <= n; i++) {
        dp[i] = dp[i - 1];
        if (i >= 3)
            dp[i] += dp[i - 3];
    }

    return dp[n];
}

int main() {
    int n;
    scanf("%d", &n);
    printf("%llu\n", countWays(n));
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 6 | 6 | 6 | ✓ |
| ✓ | 25 | 8641 | 8641 | ✓ |
| ✓ | 100 | 24382819596721629 | 24382819596721629 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.



# 2-DP-Playing with chessboard

| | |
|---|---|
| **Started on** | Tuesday, 21 October 2025, 3:46 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 21 October 2025, 3:49 PM |
| **Time taken** | 3 mins 5 secs |
| **Grade** | **10.00** out of 10.00 (**100**%) |

**Question 1** Correct Mark 10.00 out of 10.00 Flag question

**Playing with Chessboard:**

Ram is given with an n*n chessboard with each cell with a monetary value. Ram stands at the (0,0), that the position of the top left white rook. He is been given a task to reach the bottom right black rook position (n-1, n-1) constrained that he needs to reach the position by traveling the maximum monetary path under the condition that he can only travel one step right or one step down the board. Help ram to achieve it by providing an efficient DP algorithm.

**Example:**

**Input**

3

**1** 2 4

**2** 3 4

**8 7 1**

**Output:**

19

**Explanation:**

Totally there will be 6 paths among that the optimal is

Optimal path value:1+2+8+7+1=19

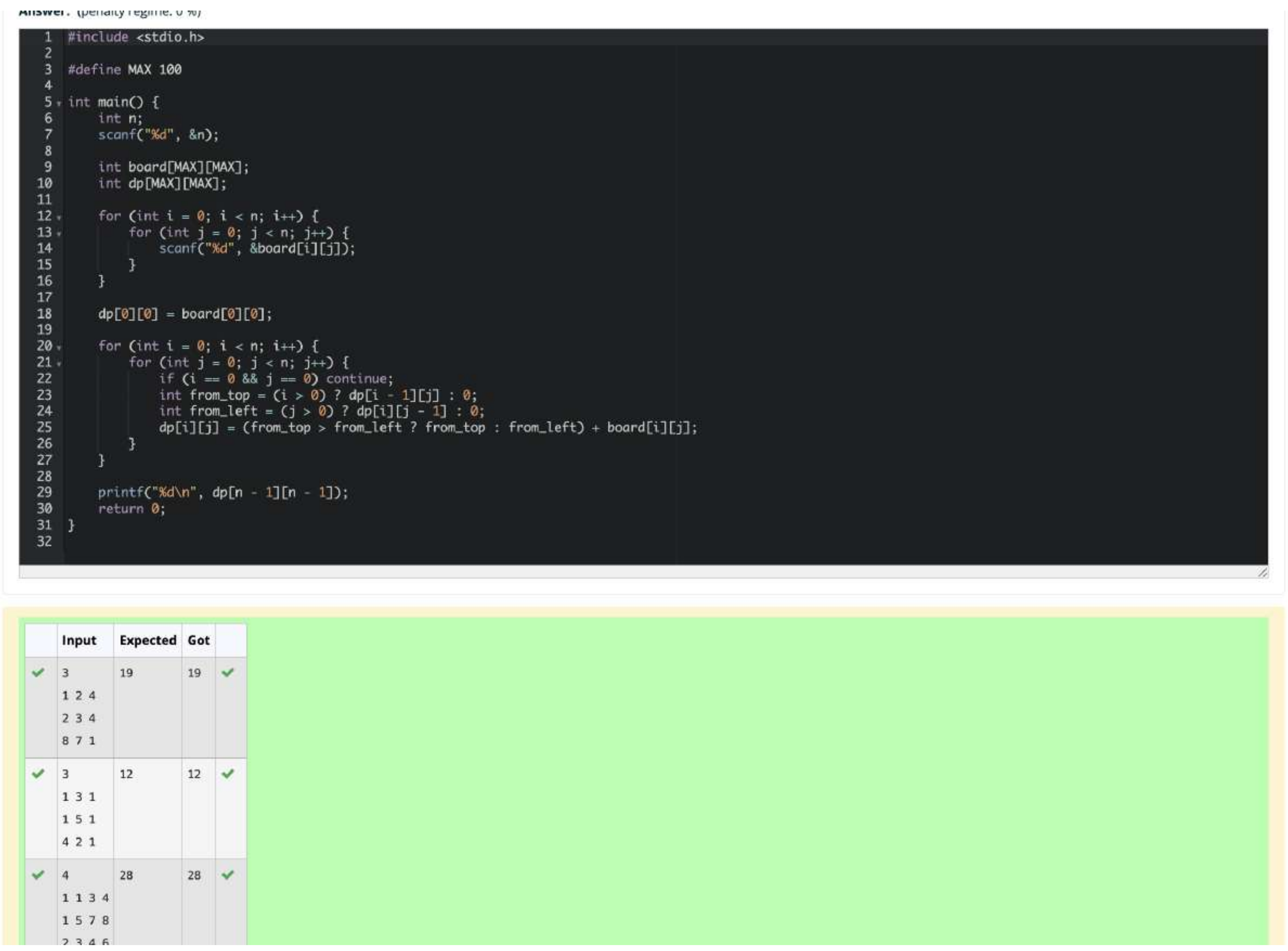
**Input Format**

First Line contains the integer n

The next n lines contain the n*n chessboard values

**Output Format**

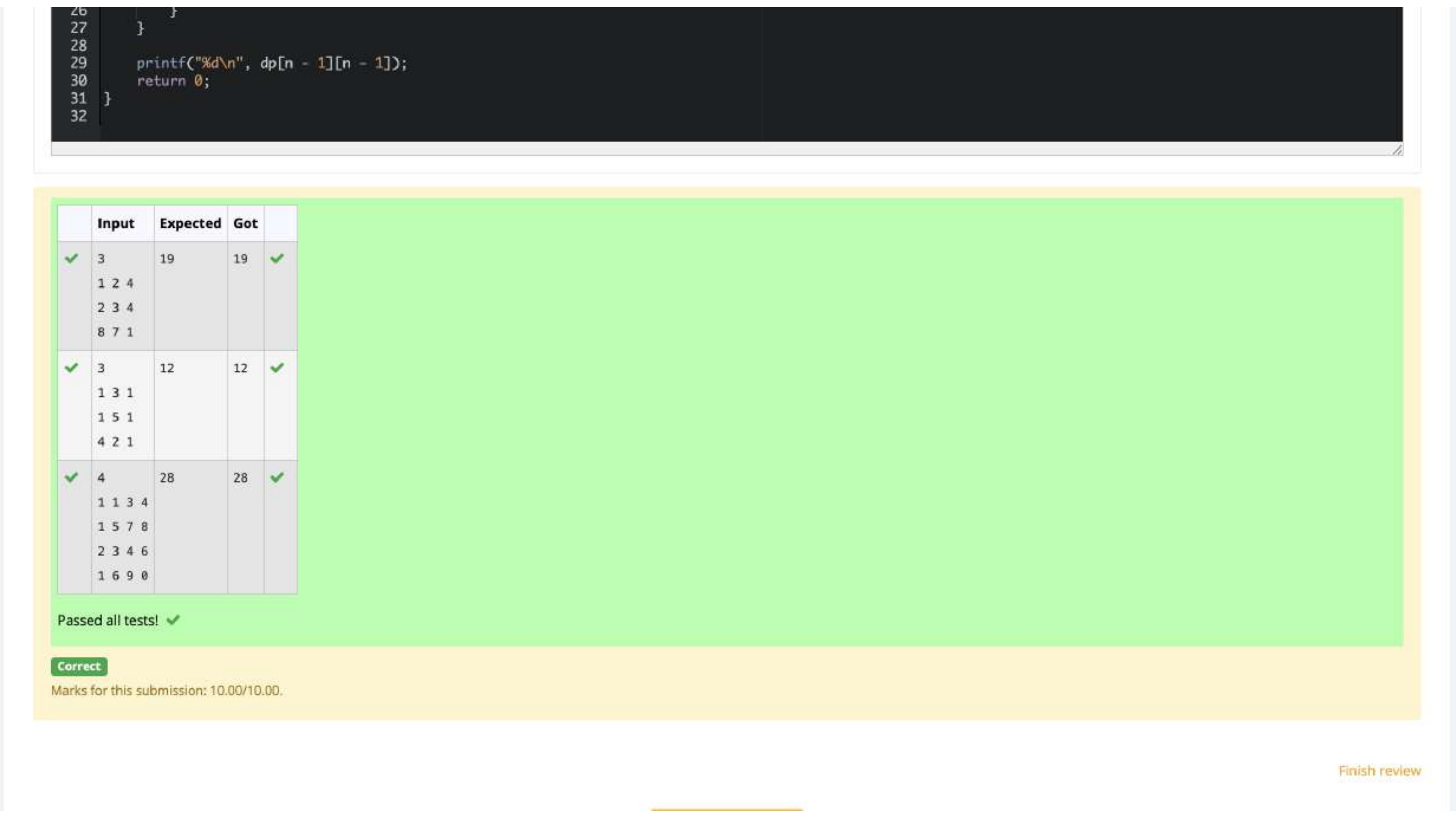
Print Maximum monetary value of the path



**Answer:** (penalty regime: 0 %)

```c
#include <stdio.h>

#define MAX 100

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

    int board[MAX][MAX];
    int dp[MAX][MAX];

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

    dp[0][0] = board[0][0];

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i == 0 && j == 0) continue;
            int from_top = (i > 0) ? dp[i - 1][j] : 0;
            int from_left = (j > 0) ? dp[i][j - 1] : 0;
            dp[i][j] = (from_top > from_left ? from_top : from_left) + board[i][j];
        }
    }

    printf("%d\n", dp[n - 1][n - 1]);
    return 0;
}

```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 3<br>1 2 4<br>2 3 4<br>8 7 1 | 19 | 19 | ✓ |
| ✓ | 3<br>1 3 1<br>1 5 1<br>4 2 1 | 12 | 12 | ✓ |
| ✓ | 4<br>1 1 3 4<br>1 5 7 8<br>2 3 4 6 | 28 | 28 | ✓ |



```
        }
    }

    printf("%d\n", dp[n - 1][n - 1]);
    return 0;
}

```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 3<br>1 2 4<br>2 3 4<br>8 7 1 | 19 | 19 | ✓ |
| ✓ | 3<br>1 3 1<br>1 5 1<br>4 2 1 | 12 | 12 | ✓ |
| ✓ | 4<br>1 1 3 4<br>1 5 7 8<br>2 3 4 6<br>1 6 9 0 | 28 | 28 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

Finish review

# 3-DP-Longest Common Subsequence

| | |
|---|---|
| **Started on** | Tuesday, 21 October 2025, 3:49 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 21 October 2025, 3:50 PM |
| **Time taken** | 46 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 two strings find the length of the common longest subsequence(need not be contiguous) between the two.

Example:

s1: ggtabe

s2: tgatasb

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| s1 | a | g | g | t | a | b | |
| s2 | g | x | t | x | a | y | b |

**The length is 4**

Solveing it using Dynamic Programming

**For example:**

| Input | Result |
|---|---|
| aab<br>azb | 2 |

**Answer:** (penalty regime: 0 %)

**Answer:** (penalty regime: 0 %)

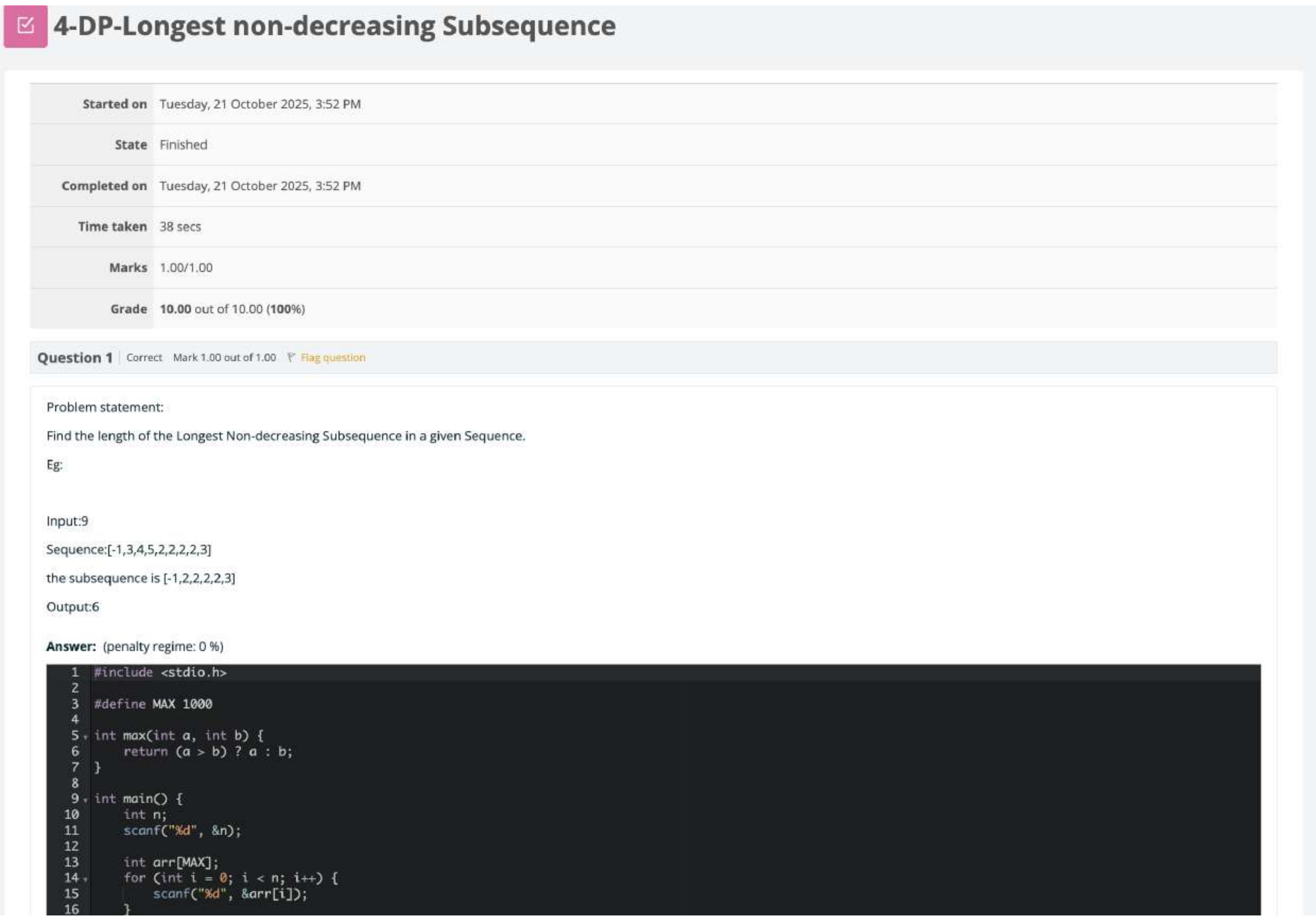
```c
#include <stdio.h>
#include <string.h>

#define MAX 1000

int max(int a, int b) {
    return (a > b) ? a : b;
}

int main() {
    char s1[MAX], s2[MAX];
    scanf("%s %s", s1, s2);

    int len1 = strlen(s1);
    int len2 = strlen(s2);

    int dp[MAX][MAX];

    for (int i = 0; i <= len1; i++) {
        for (int j = 0; j <= len2; j++) {
            if (i == 0 || j == 0)
                dp[i][j] = 0;
            else if (s1[i - 1] == s2[j - 1])
                dp[i][j] = dp[i - 1][j - 1] + 1;
            else
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
        }
    }

    printf("%d\n", dp[len1][len2]);
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | aab<br>azb | 2 | 2 | ✓ |
| ✓ | ABCD<br>ABCD | 4 | 4 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



# 4-DP-Longest non-decreasing Subsequence

| | |
|---|---|
| **Started on** | Tuesday, 21 October 2025, 3:52 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 21 October 2025, 3:52 PM |
| **Time taken** | 38 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:

Find the length of the Longest Non-decreasing Subsequence in a given Sequence.

Eg:

Input:9

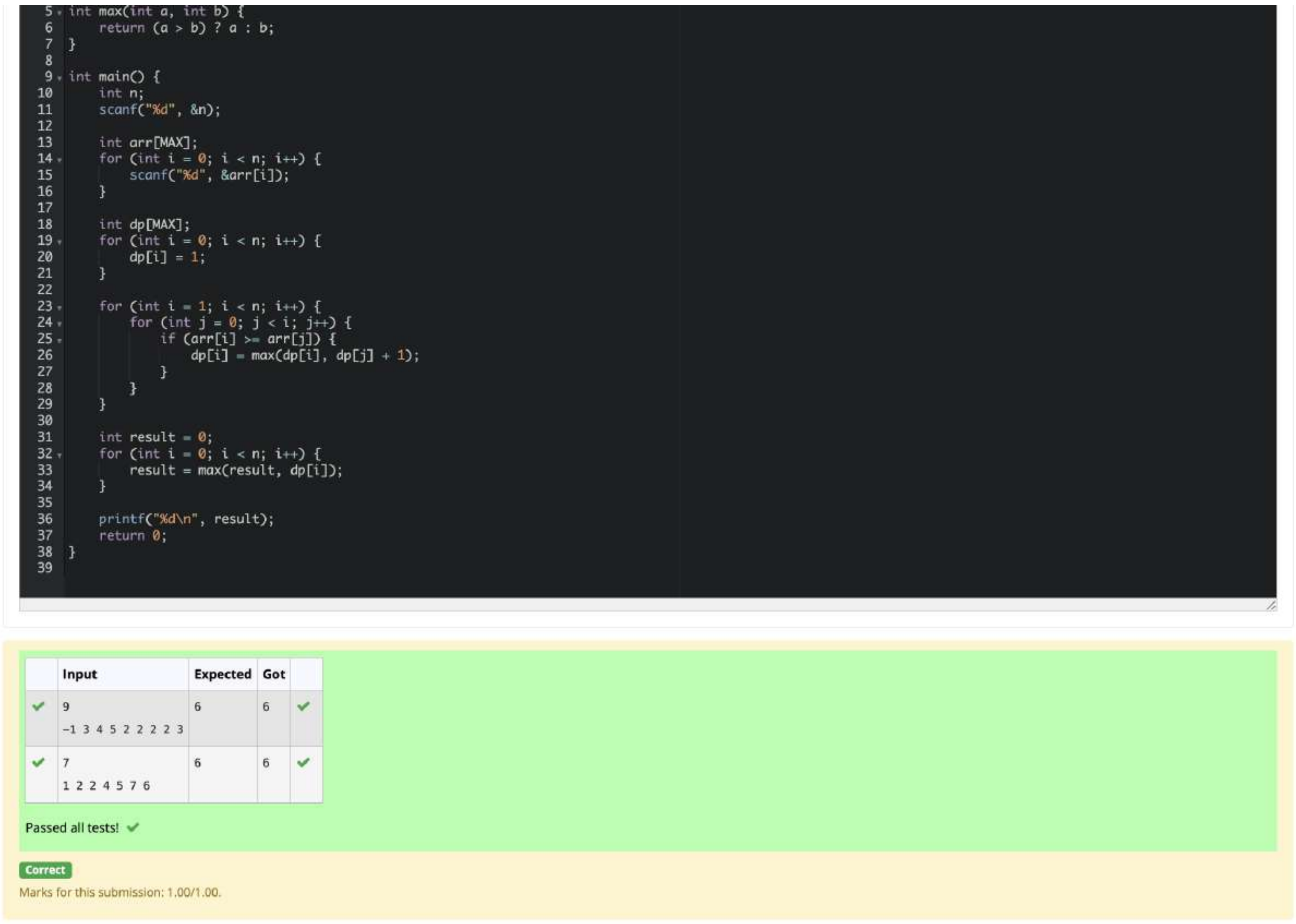
Sequence:[-1,3,4,5,2,2,2,2,3]

the subsequence is [-1,2,2,2,2,3]

Output:6

**Answer:** (penalty regime: 0 %)

```c
#include <stdio.h>

#define MAX 1000

int max(int a, int b) {
    return (a > b) ? a : b;
}

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

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



```c
int max(int a, int b) {
    return (a > b) ? a : b;
}

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

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

    int dp[MAX];
    for (int i = 0; i < n; i++) {
        dp[i] = 1;
    }

    for (int i = 1; i < n; i++) {
        for (int j = 0; j < i; j++) {
            if (arr[i] >= arr[j]) {
                dp[i] = max(dp[i], dp[j] + 1);
            }
        }
    }

    int result = 0;
    for (int i = 0; i < n; i++) {
        result = max(result, dp[i]);
    }

    printf("%d\n", result);
    return 0;
}

```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 9<br>-1 3 4 5 2 2 2 2 3 | 6 | 6 | ✓ |
| ✓ | 7<br>1 2 2 4 5 7 6 | 6 | 6 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.