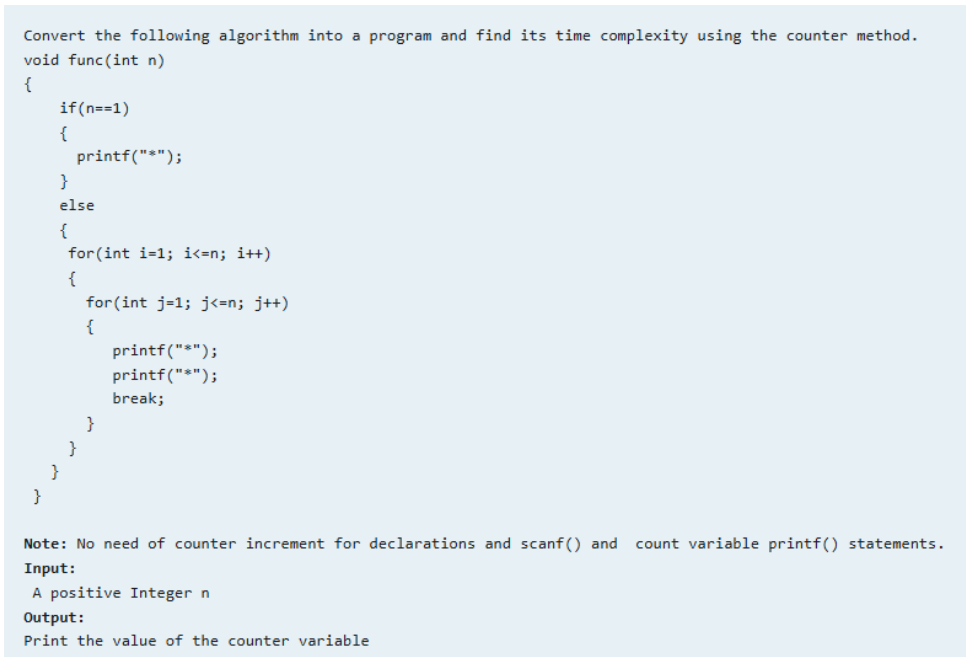
DESIGN AND ANALYSIS OF ALGORITHMS

HARINI M
230701101
CSE B SECTION

FINDING TIME COMPLEXITIES OF ALGORITHMS



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;
       }
     }
   }
 }
```

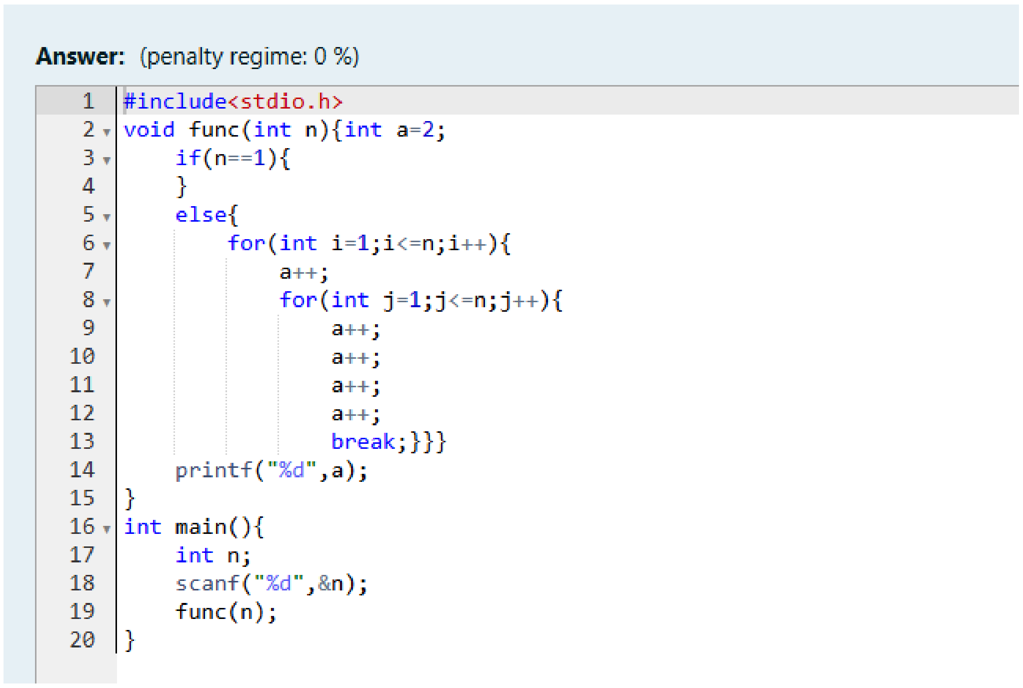
Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:

 A positive Integer n

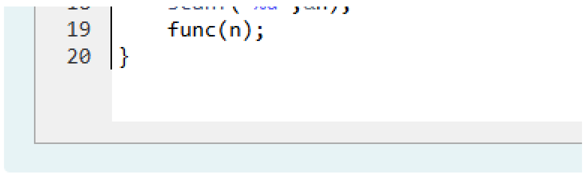
Output:

Print the value of the counter variable

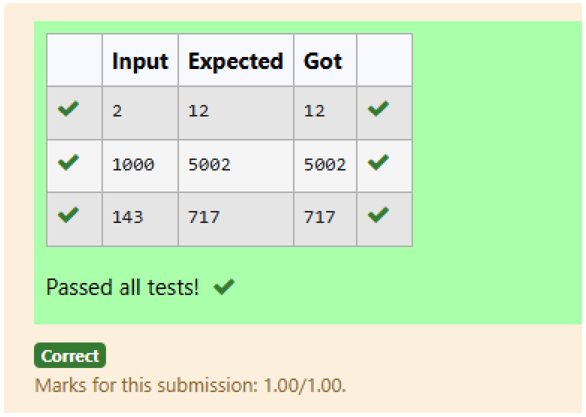


Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
void func(int n){int a=2;
    if(n==1){
    }
    else{
        for(int i=1;i<=n;i++){
            a++;
            for(int j=1;j<=n;j++){
                a++;
                a++;
                a++;
                a++;
                break;}}}
    printf("%d",a);
}
int main(){
    int n;
    scanf("%d",&n);
    func(n);
}
```



```
    func(n);
}
```

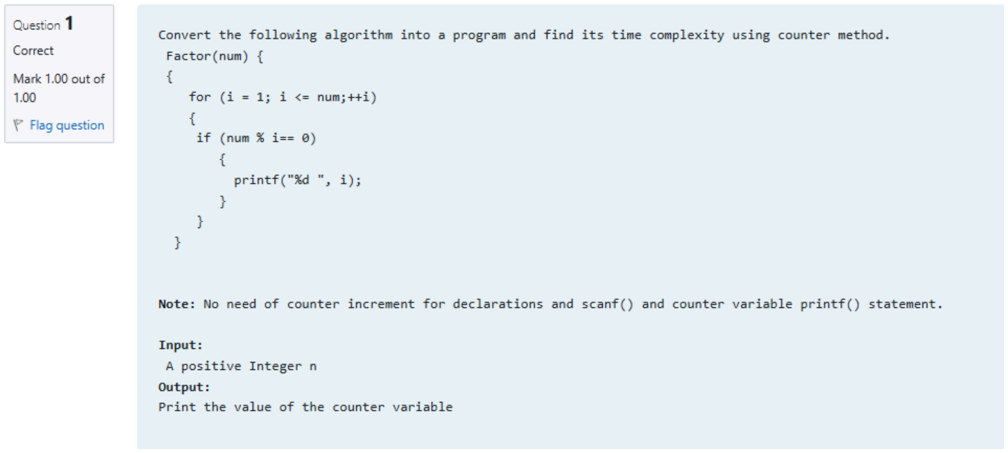


	Input	Expected	Got	
✓	2	12	12	✓
✓	1000	5002	5002	✓
✓	143	717	717	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



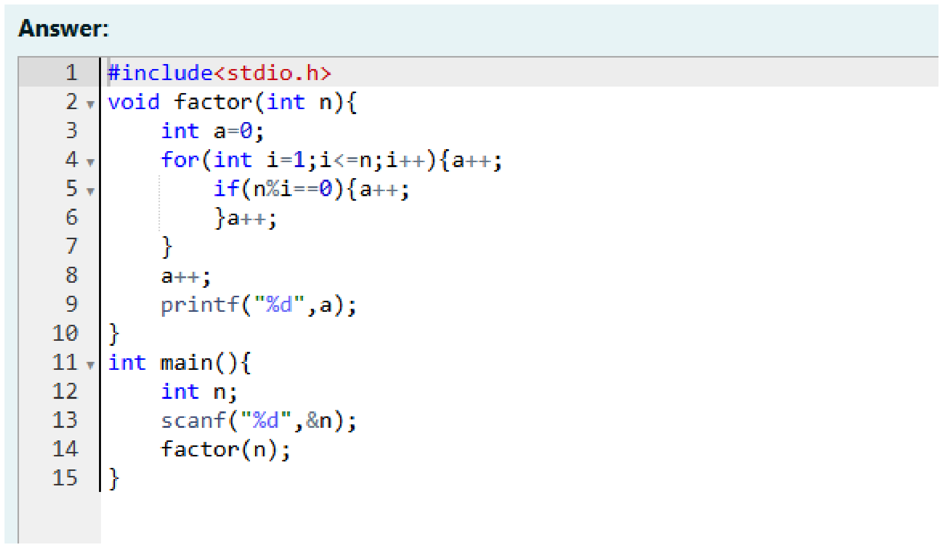
Question **1**

Correct

Mark 1.00 out of 1.00

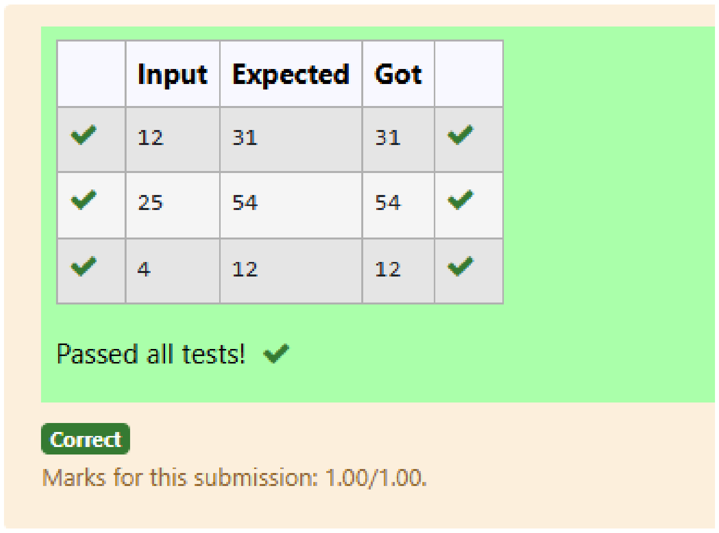
Flag question

```
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
```



Answer:

```c
#include<stdio.h>
void factor(int n){
    int a=0;
    for(int i=1;i<=n;i++){a++;
        if(n%i==0){a++;
        }a++;
    }
    a++;
    printf("%d",a);
}
int main(){
    int n;
    scanf("%d",&n);
    factor(n);
}
```

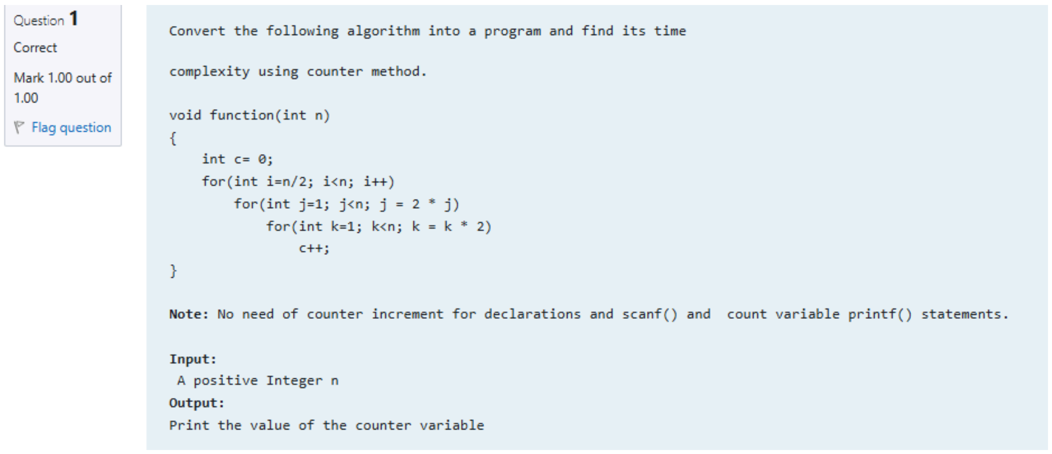


	Input	Expected	Got	
✓	12	31	31	✓
✓	25	54	54	✓
✓	4	12	12	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



Question **1**

Correct

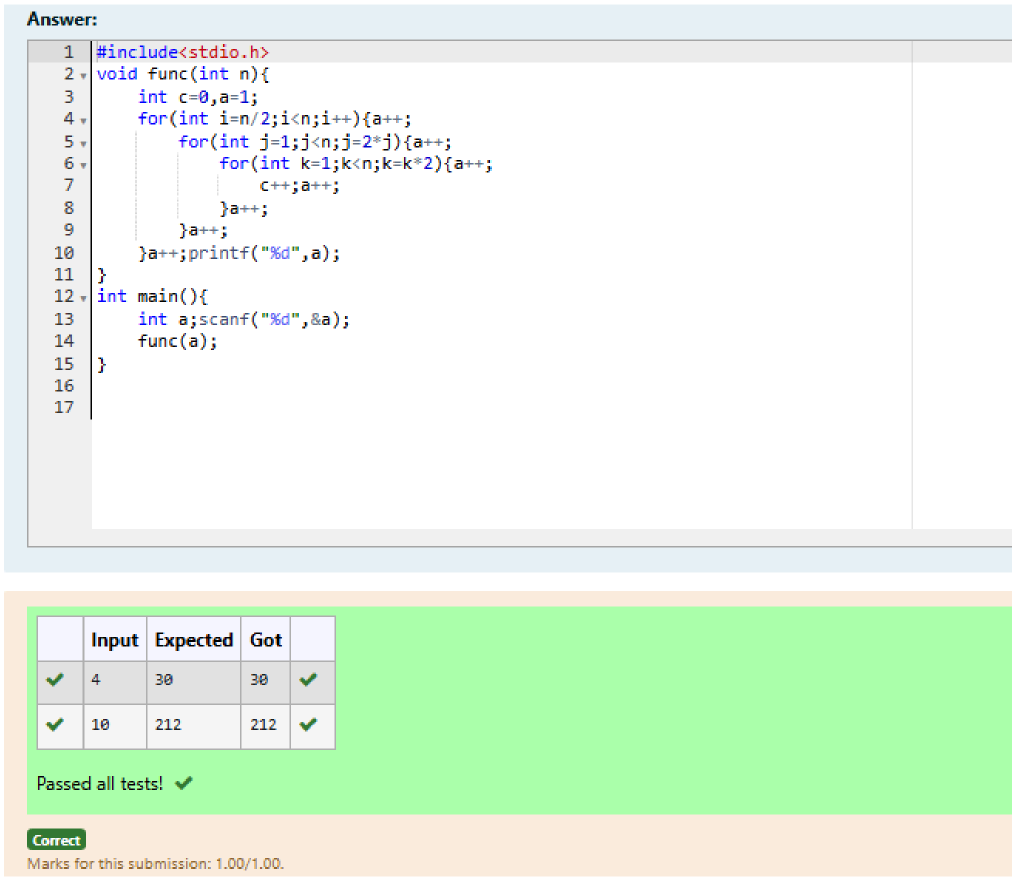
Mark 1.00 out of 1.00

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:

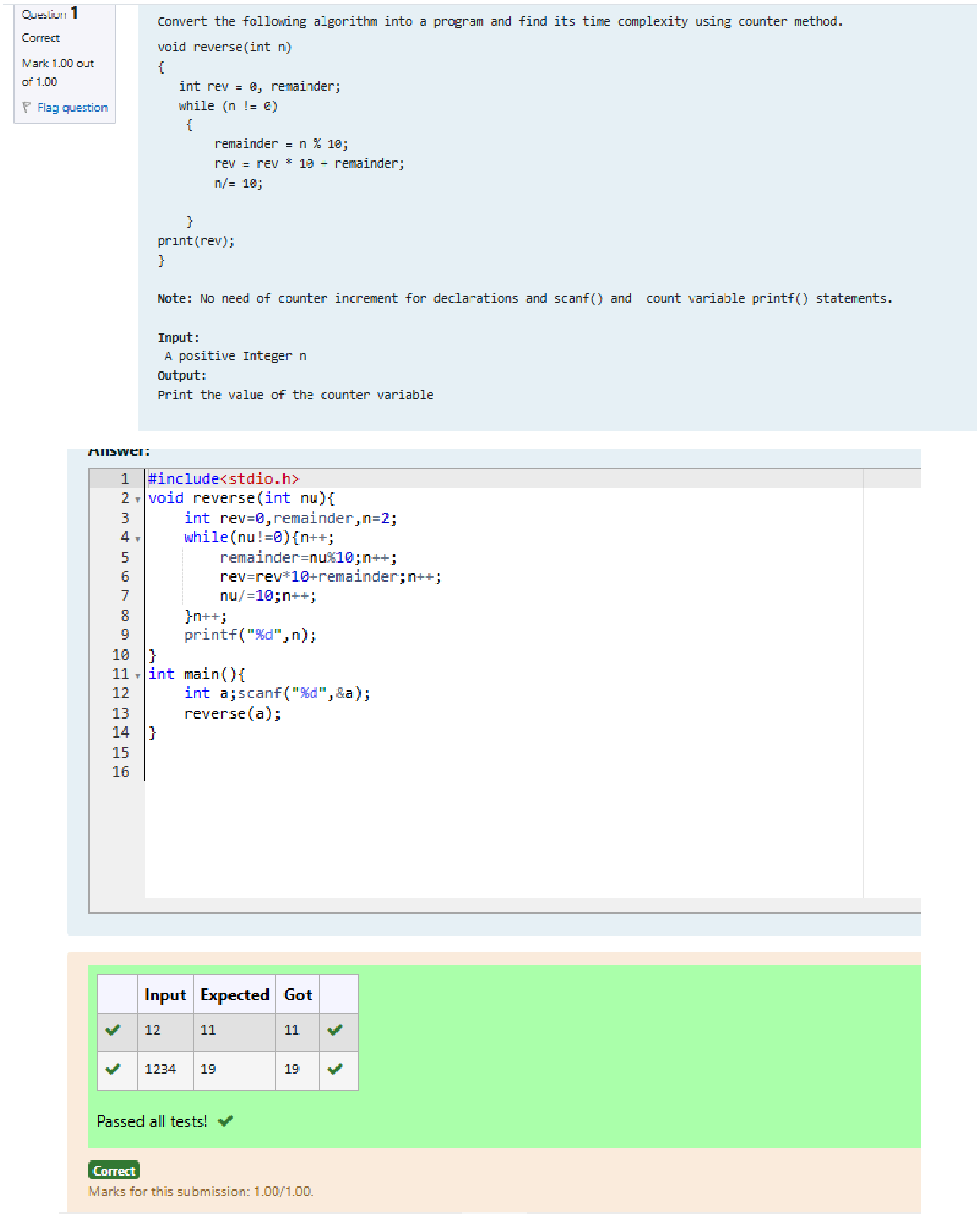
```c
#include<stdio.h>
void func(int n){
    int c=0,a=1;
    for(int i=n/2;i<n;i++){a++;
        for(int j=1;j<n;j=2*j){a++;
            for(int k=1;k<n;k=k*2){a++;
                c++;a++;
            }a++;
        }a++;
    }a++;printf("%d",a);
}
int main(){
    int a;scanf("%d",&a);
    func(a);
}
```

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

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.


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

Answer:

```
#include<stdio.h>
void reverse(int nu){
    int rev=0,remainder,n=2;
    while(nu!=0){n++;
        remainder=nu%10;n++;
        rev=rev*10+remainder;n++;
        nu/=10;n++;
    }n++;
    printf("%d",n);
}
int main(){
    int a;scanf("%d",&a);
    reverse(a);
}
```

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

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

SOLVING OF GREEDY ALGORITHM

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 %)

```c
#include <stdio.h>
int main() {
    int V;
    scanf("%d", &V);
    int denomination[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
    int num_denomination = sizeof(denomination) / sizeof(denomination[0]);
    int count = 0;
    for (int i = 0; i < num_denomination; i++) {
        count += V / denomination[i];
        V %= denomination[i];
    }
    printf("%d\n", count);
    return 0;
}
```

	Input	Expected	Got	
✓	49	5	5	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

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.

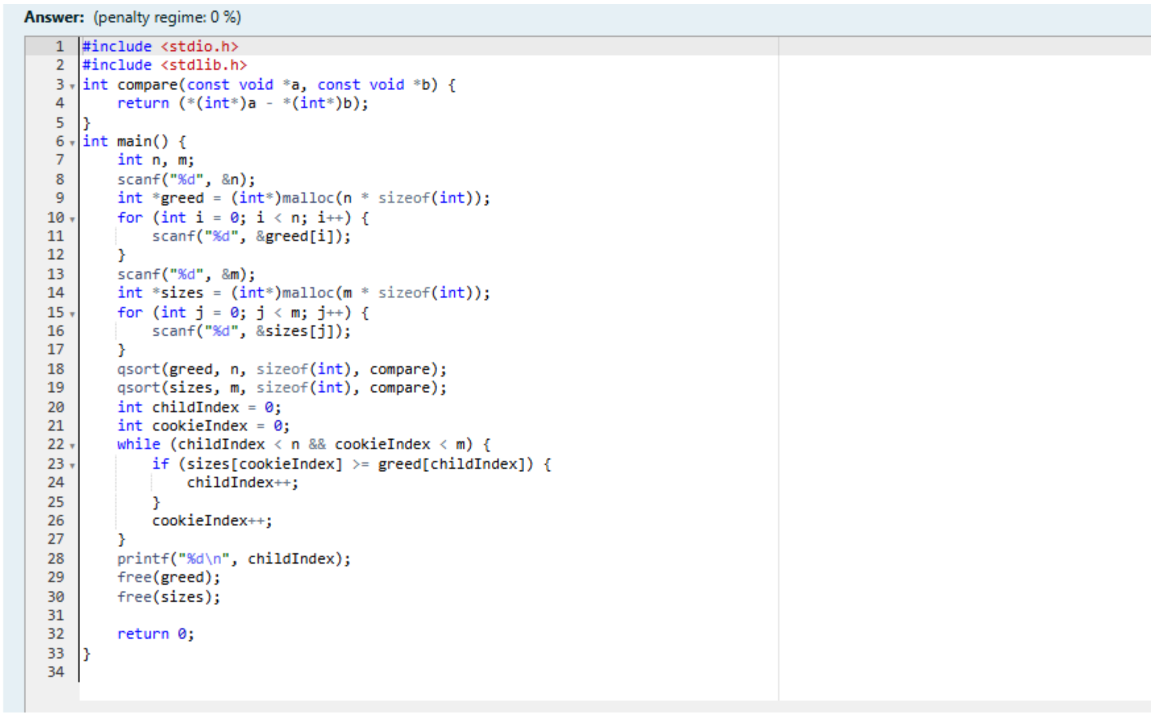
You need to output 1.

Constraints:

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

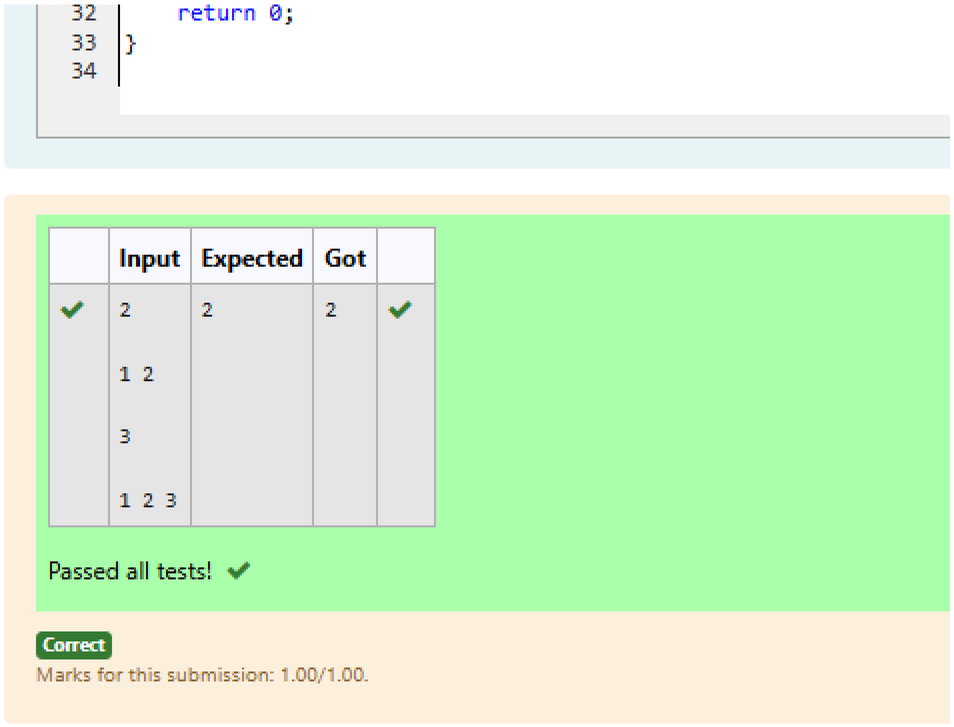
0 <= s.length <= 3 * 10^4

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



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, m;
    scanf("%d", &n);
    int *greed = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &greed[i]);
    }
    scanf("%d", &m);
    int *sizes = (int*)malloc(m * sizeof(int));
    for (int j = 0; j < m; j++) {
        scanf("%d", &sizes[j]);
    }
    qsort(greed, n, sizeof(int), compare);
    qsort(sizes, m, sizeof(int), compare);
    int childIndex = 0;
    int cookieIndex = 0;
    while (childIndex < n && cookieIndex < m) {
        if (sizes[cookieIndex] >= greed[childIndex]) {
            childIndex++;
        }
        cookieIndex++;
    }
    printf("%d\n", childIndex);
    free(greed);
    free(sizes);

    return 0;
}
```



```c
    return 0;
}
```

	Input	Expected	Got	
✓	2 1 2 3 1 2 3	2	2	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

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 calorie
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
```

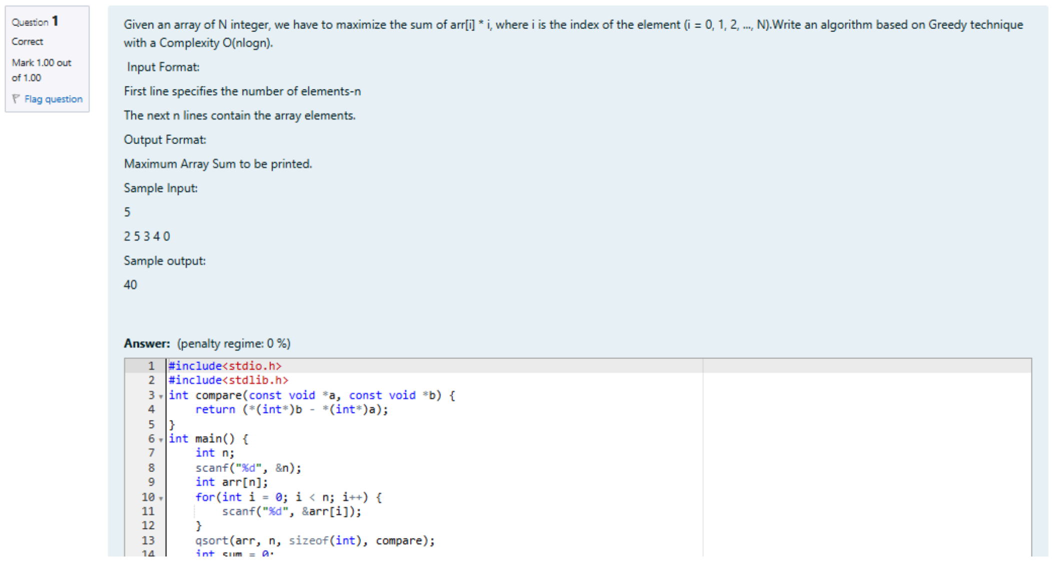
For example:

Test	Input	Result
Test Case 1	3 1 3 2	18

```c
#include<stdio.h>
#include<math.h>
int main(){
    int a;scanf("%d",&a);int arr[a],sum=0;
    for(int i=0;i<a;i++)scanf("%d",&arr[i]);
    for(int i=0;i<a-1;i++){
        for(int j=i;j<a;j++){
            if(arr[i]<arr[j]){
                int temp=arr[i];arr[i]=arr[j];arr[j]=temp;

            }
        }
    }
    for(int i=0;i<a;i++)sum+=pow(a,i)*arr[i];
    printf("%d",sum);
}
```

	Test	Input	Expected	Got	
✓	Test Case 1	3 1 3 2	18	18	✓
✓	Test Case 2	4 7 4 9 6	389	389	✓
✓	Test Case 3	3 5 10 7	76	76	✓

Passed all tests! ✓



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

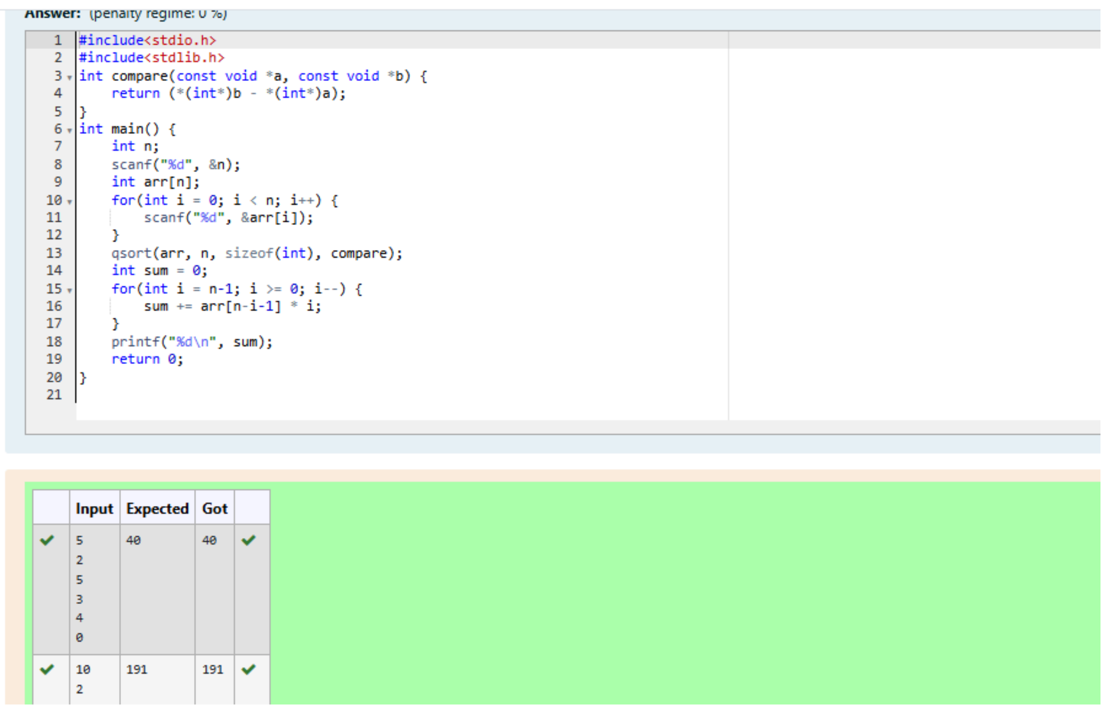
2 5 3 4 0

Sample output:

40

Answer: (penalty regime: 0 %)

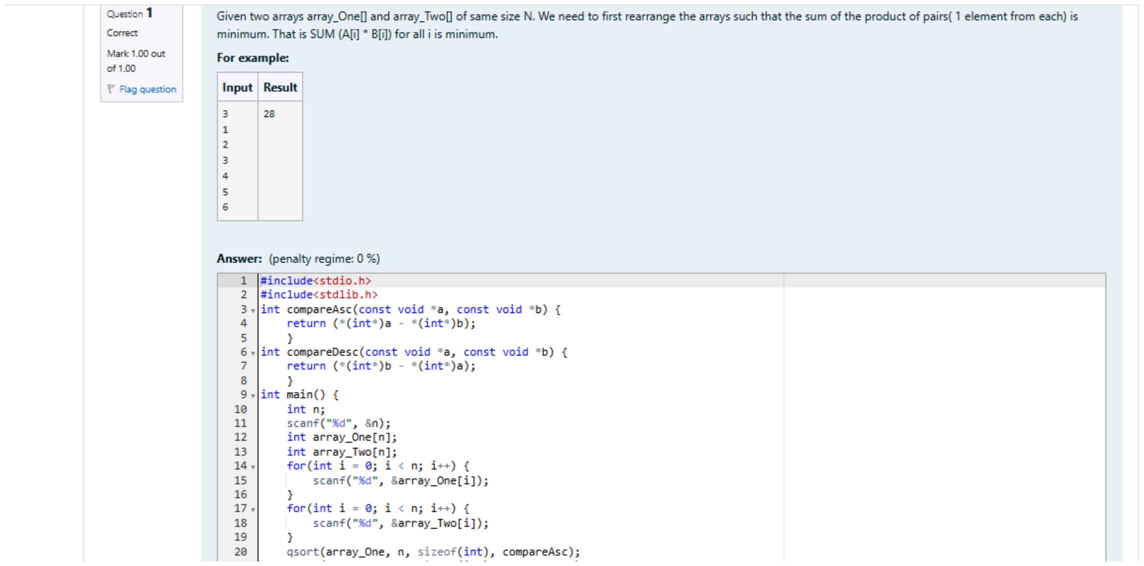
```c
#include<stdio.h>
#include<stdlib.h>
int compare(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
}
int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for(int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    qsort(arr, n, sizeof(int), compare);
    int sum = 0;
```



Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
#include<stdlib.h>
int compare(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
}
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);
    int sum = 0;
    for(int i = n-1; i >= 0; i--) {
        sum += arr[n-i-1] * i;
    }
    printf("%d\n", sum);
    return 0;
}
```

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



Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

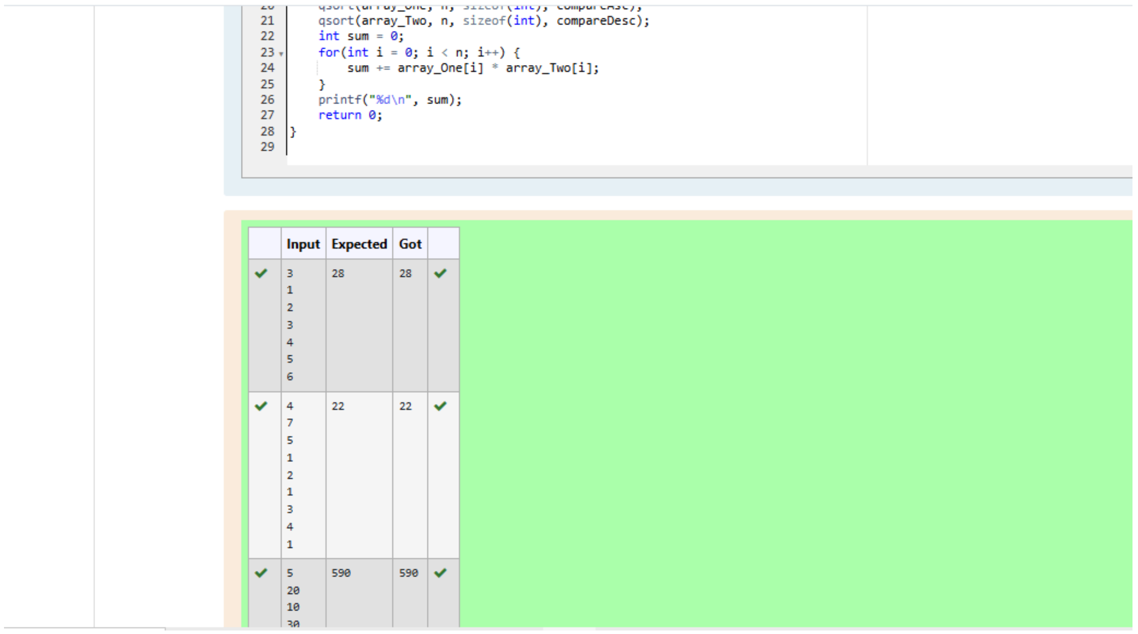
Given two arrays array_One[] and array_Two[] of same size N. We need to first rearrange the arrays such that the sum of the product of pairs(1 element from each) is minimum. That is SUM (A[i] * B[i]) for all i is minimum.

For example:

Input	Result
3 1 2 3 4 5 6	28

Answer: (penalty regime: 0 %)

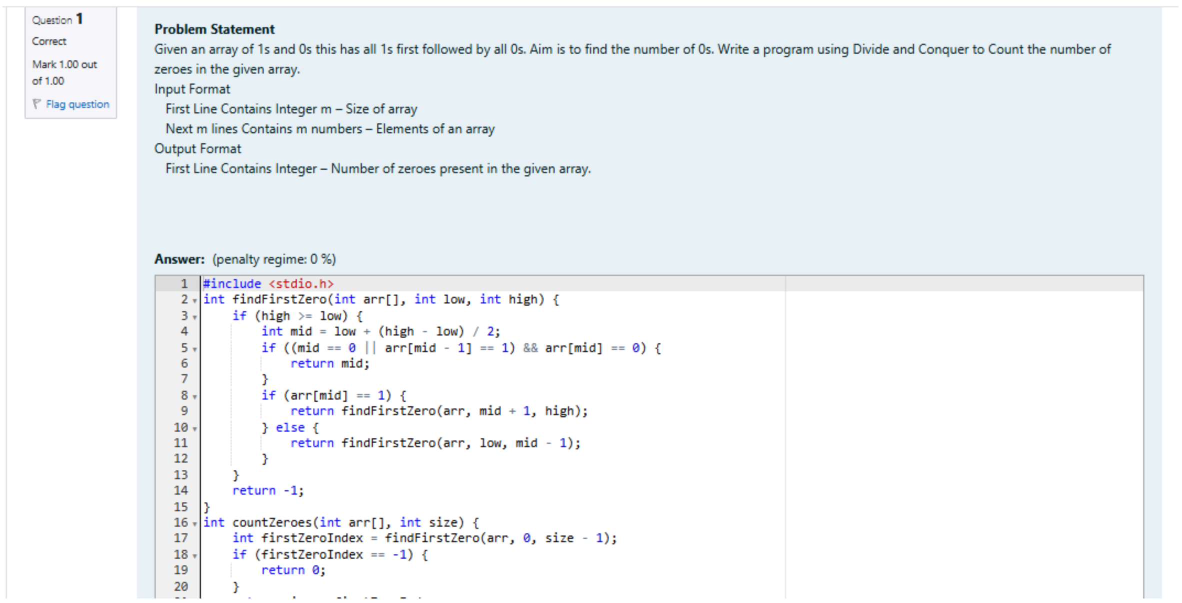
```c
#include<stdio.h>
#include<stdlib.h>
int compareAsc(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
    }
int compareDesc(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
    }
int main() {
    int n;
    scanf("%d", &n);
    int array_One[n];
    int array_Two[n];
    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), compareAsc);
```



```c
    qsort(array_Two, n, sizeof(int), compareDesc);
    int sum = 0;
    for(int i = 0; i < n; i++) {
        sum += array_One[i] * array_Two[i];
    }
    printf("%d\n", sum);
    return 0;
}
```

	Input	Expected	Got	
✓	3 1 2 3 4 5 6	28	28	✓
✓	4 7 5 1 2 1 3 4 1	22	22	✓
✓	5 20 10 30	590	590	✓

PROBLEMS IN DIVIDE AND CONQUER TECHNIQUE



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

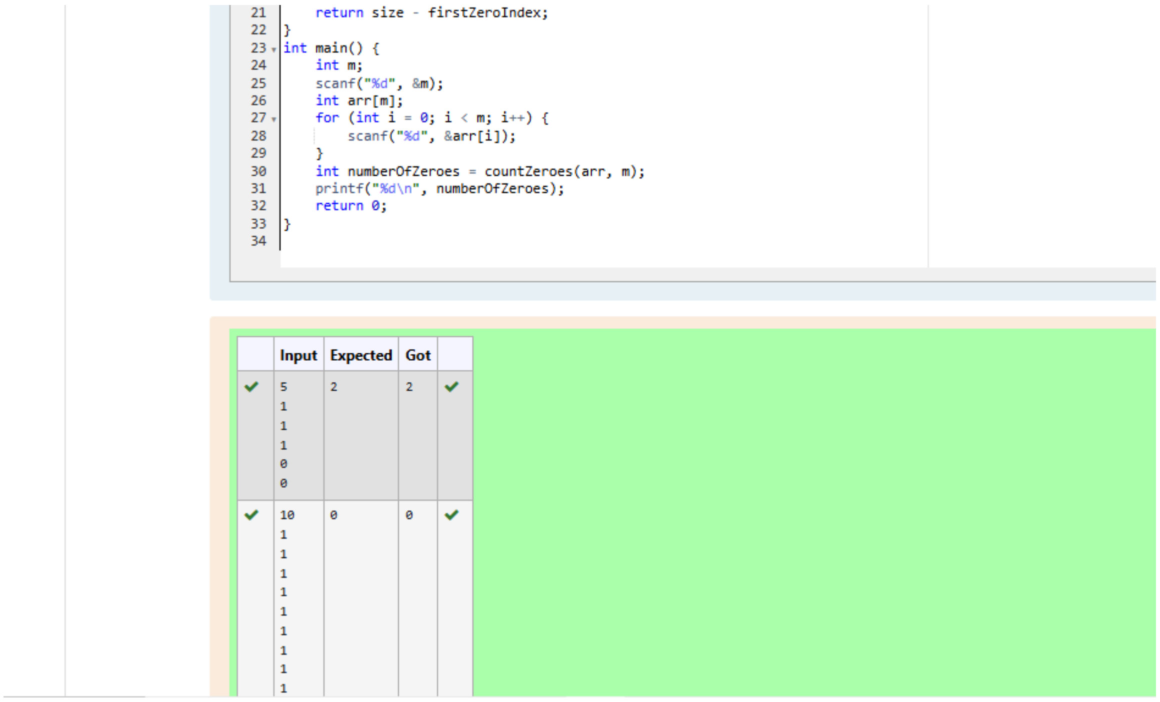
Next m lines Contains m numbers – Elements of an array

Output Format

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

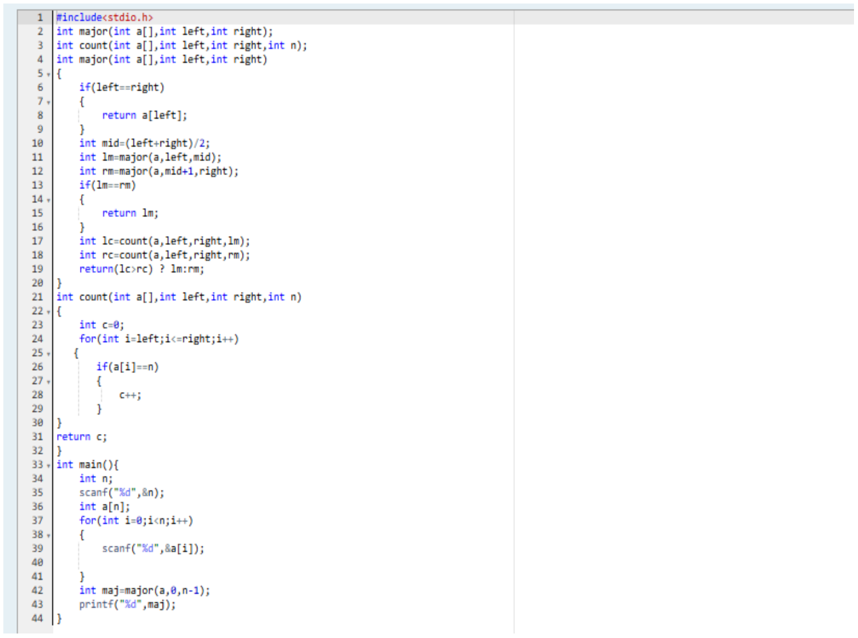
Answer: (penalty regime: 0 %)

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

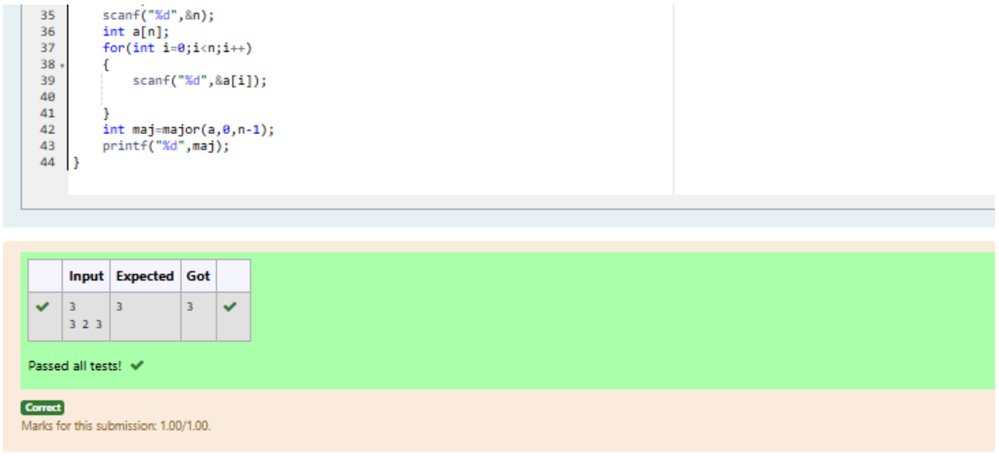


```c
    return size - firstZeroIndex;
}
int main() {
    int m;
    scanf("%d", &m);
    int arr[m];
    for (int i = 0; i < m; i++) {
        scanf("%d", &arr[i]);
    }
    int numberOfZeroes = countZeroes(arr, m);
    printf("%d\n", numberOfZeroes);
    return 0;
}
```

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



```c
#include<stdio.h>
int major(int a[],int left,int right);
int count(int a[],int left,int right,int n);
int major(int a[],int left,int right)
{
    if(left==right)
    {
        return a[left];
    }
    int mid=(left+right)/2;
    int lm=major(a,left,mid);
    int rm=major(a,mid+1,right);
    if(lm==rm)
    {
        return lm;
    }
    int lc=count(a,left,right,lm);
    int rc=count(a,left,right,rm);
    return(lc>rc) ? lm:rm;
}
int count(int a[],int left,int right,int n)
{
    int c=0;
    for(int i=left;i<=right;i++)
   {
       if(a[i]==n)
       {
           c++;
       }
}
return c;
}
int main(){
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);

    }
    int maj=major(a,0,n-1);
    printf("%d",maj);
}
```



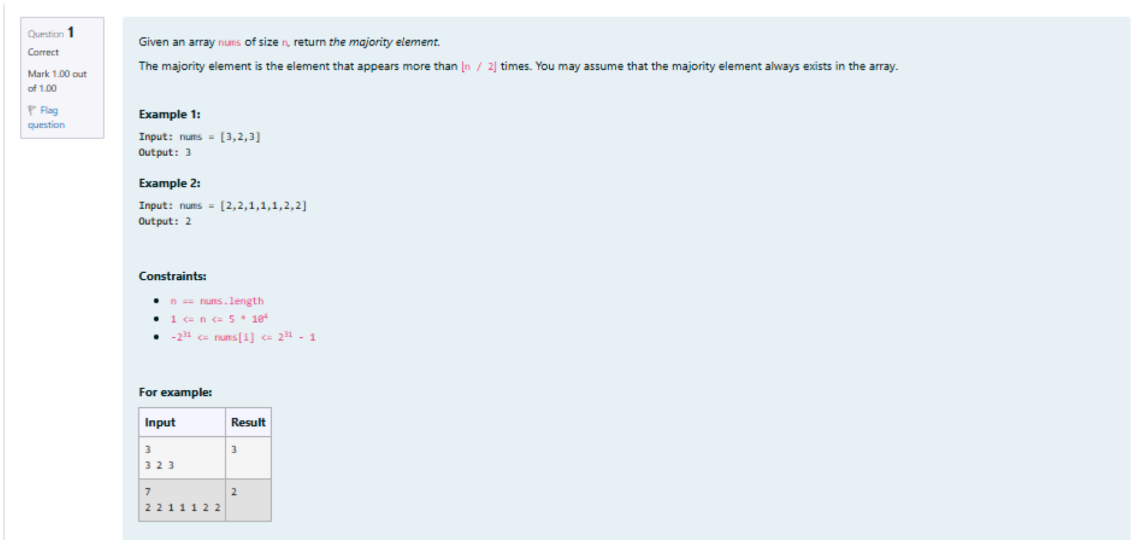
```c
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);

    }
    int maj=major(a,0,n-1);
    printf("%d",maj);
}
```

	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

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 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:

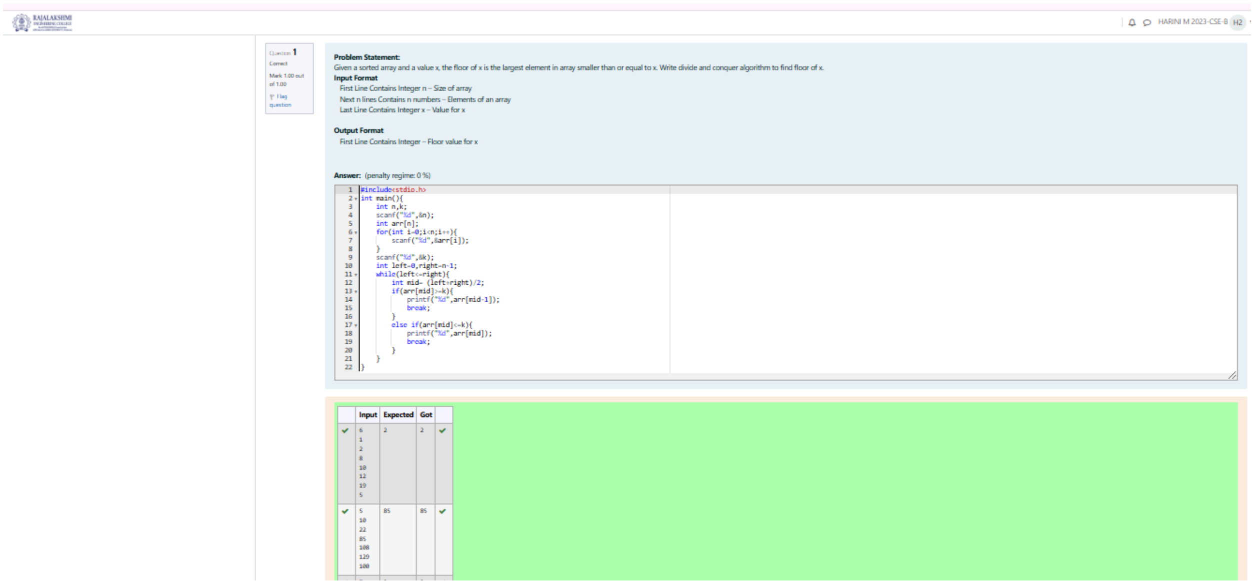
```
Input: nums = [2,2,1,1,1,2,2]
Output: 2
```

Constraints:

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

For example:

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



RAJALAKSHMI ENGINEERING COLLEGE

HARINI M 2023-CSE-B H2

Question 1
Correct
Mark 1.00 out of 1.00
Flag question

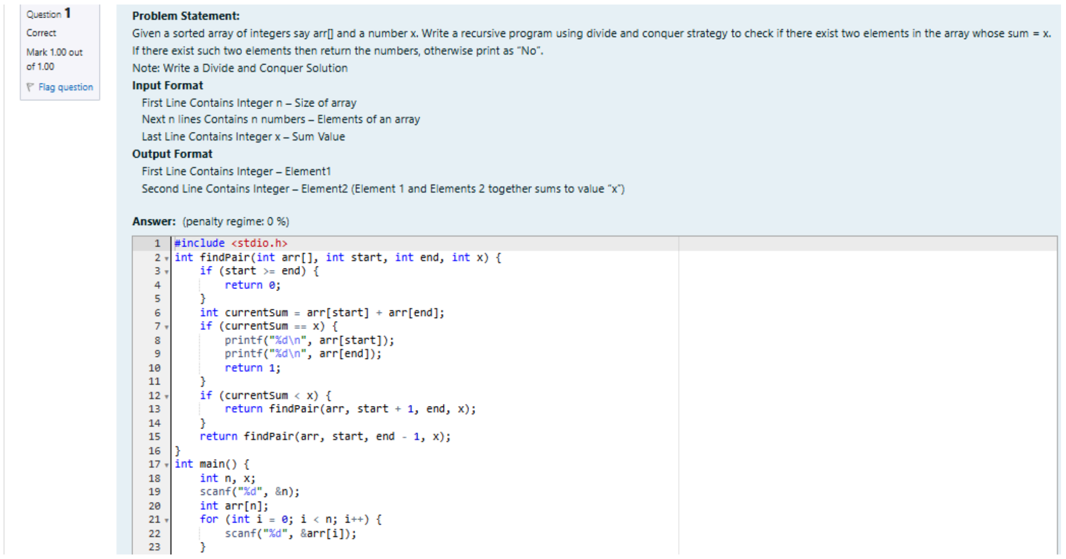
Problem Statement:
Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write divide and conquer algorithm to find floor of x.
Input Format
First Line Contains Integer n – Size of array
Next n lines Contains n numbers – Elements of an array
Last 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 main(){
    int n,k;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    scanf("%d",&k);
    int left=0,right=n-1;
    while(left<=right){
        int mid= (left+right)/2;
        if(arr[mid]>=k){
            printf("%d",arr[mid-1]);
            break;
        }
        else if(arr[mid]<=k){
            printf("%d",arr[mid]);
            break;
        }
    }
}
```

	Input	Expected	Got	
✓	6 1 2 8 10 12 19 5	2	2	✓
✓	5 10 22 85 108 129 100	85	85	✓


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

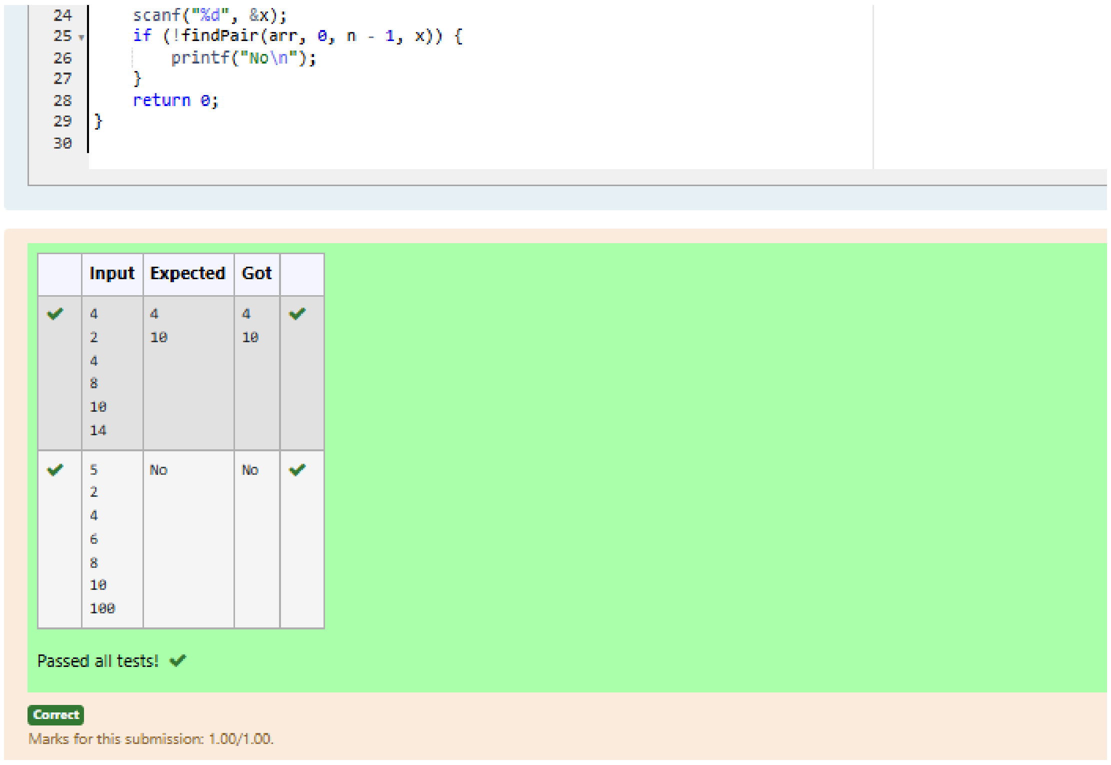
Output Format

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>
int findPair(int arr[], int start, int end, int x) {
    if (start >= end) {
        return 0;
    }
    int currentSum = arr[start] + arr[end];
    if (currentSum == x) {
        printf("%d\n", arr[start]);
        printf("%d\n", arr[end]);
        return 1;
    }
    if (currentSum < x) {
        return findPair(arr, start + 1, end, x);
    }
    return findPair(arr, start, end - 1, x);
}
int main() {
    int n, x;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
```



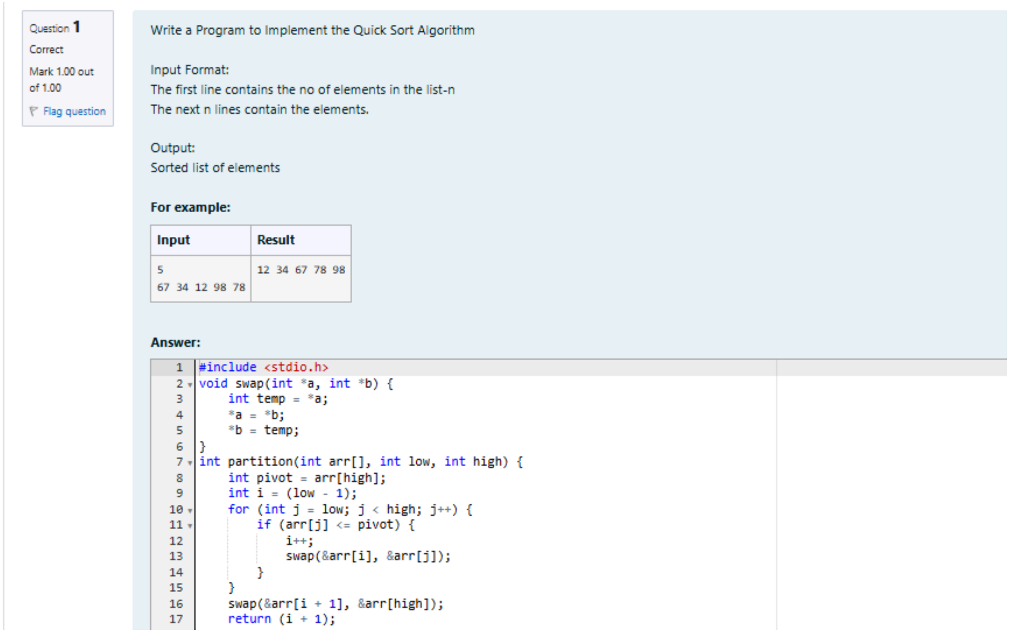
```c
    scanf("%d", &x);
    if (!findPair(arr, 0, n - 1, x)) {
        printf("No\n");
    }
    return 0;
}
```

	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.


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.

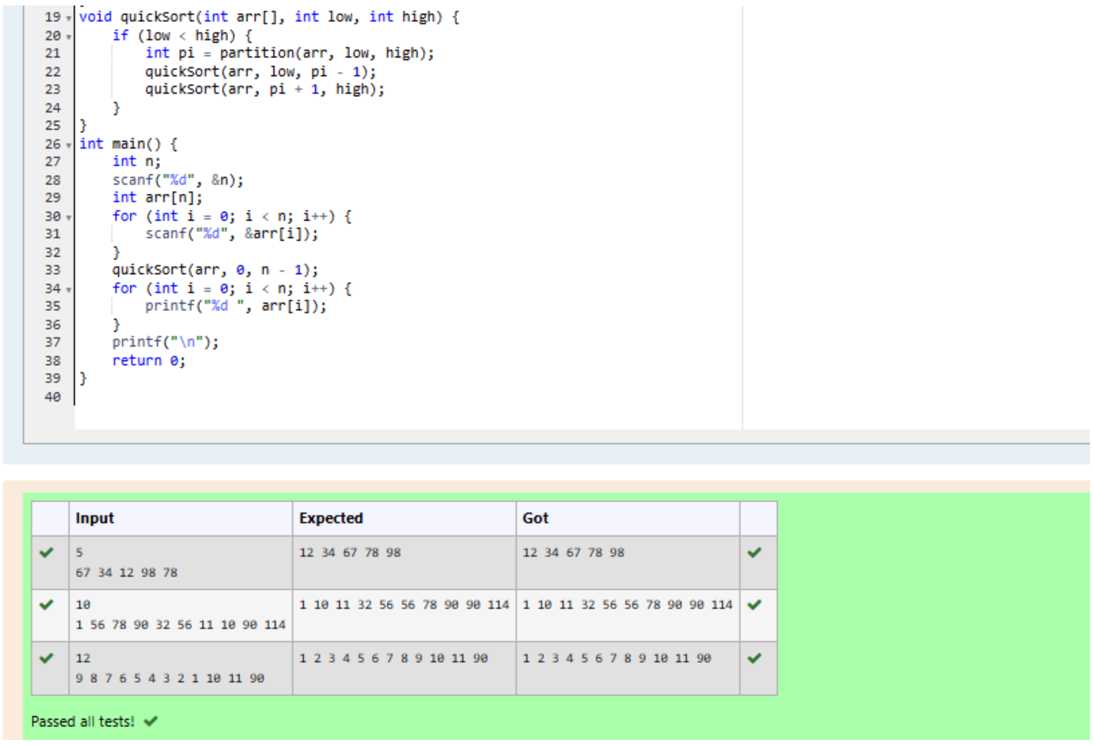
Output:
Sorted list of elements

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 temp = *a;
    *a = *b;
    *b = temp;
}
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);
```

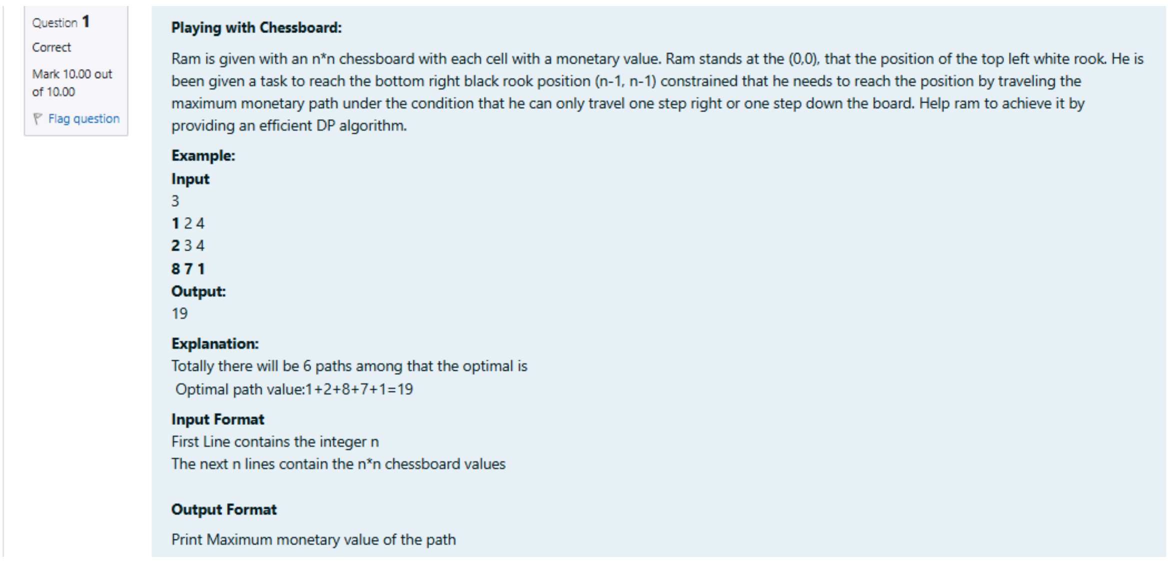


```c
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    quickSort(arr, 0, n - 1);
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    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! ✓

PROGRAMS IN DYNAMIC PROGRAMMING



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

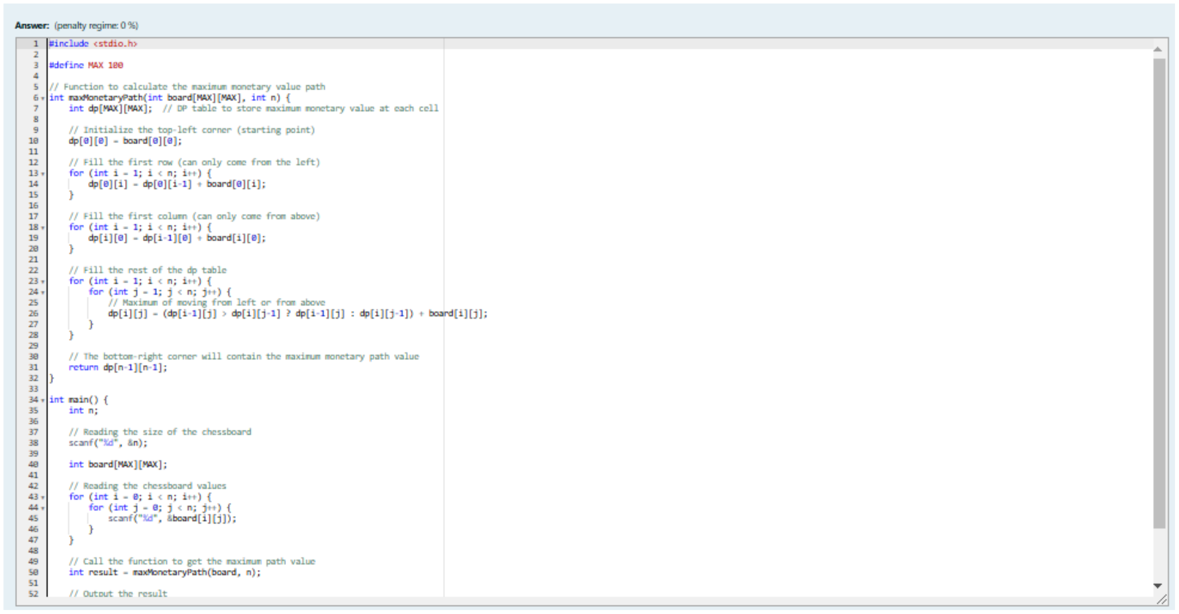
Input Format

First Line contains the integer n

The next n lines contain the n*n chessboard values

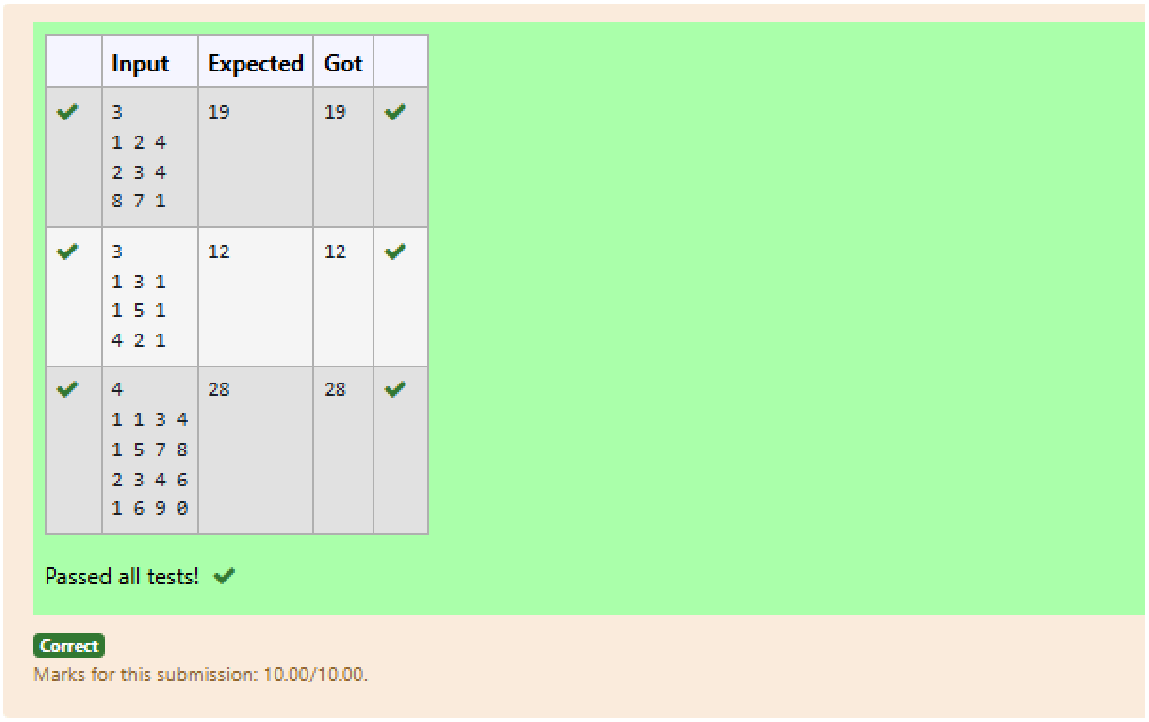
Output Format

Print Maximum monetary value of the path



Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

#define MAX 100

// Function to calculate the maximum monetary value path
int maxMonetaryPath(int board[MAX][MAX], int n) {
    int dp[MAX][MAX];  // DP table to store maximum monetary value at each cell

    // Initialize the top-left corner (starting point)
    dp[0][0] = board[0][0];

    // Fill the first row (can only come from the left)
    for (int i = 1; i < n; i++) {
        dp[0][i] = dp[0][i-1] + board[0][i];
    }

    // Fill the first column (can only come from above)
    for (int i = 1; i < n; i++) {
        dp[i][0] = dp[i-1][0] + board[i][0];
    }

    // Fill the rest of the dp table
    for (int i = 1; i < n; i++) {
        for (int j = 1; j < n; j++) {
            // Maximum of moving from left or from above
            dp[i][j] = (dp[i-1][j] > dp[i][j-1] ? dp[i-1][j] : dp[i][j-1]) + board[i][j];
        }
    }

    // The bottom-right corner will contain the maximum monetary path value
    return dp[n-1][n-1];
}

int main() {
    int n;

    // Reading the size of the chessboard
    scanf("%d", &n);

    int board[MAX][MAX];

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

    // Call the function to get the maximum path value
    int result = maxMonetaryPath(board, n);

    // Output the result
```

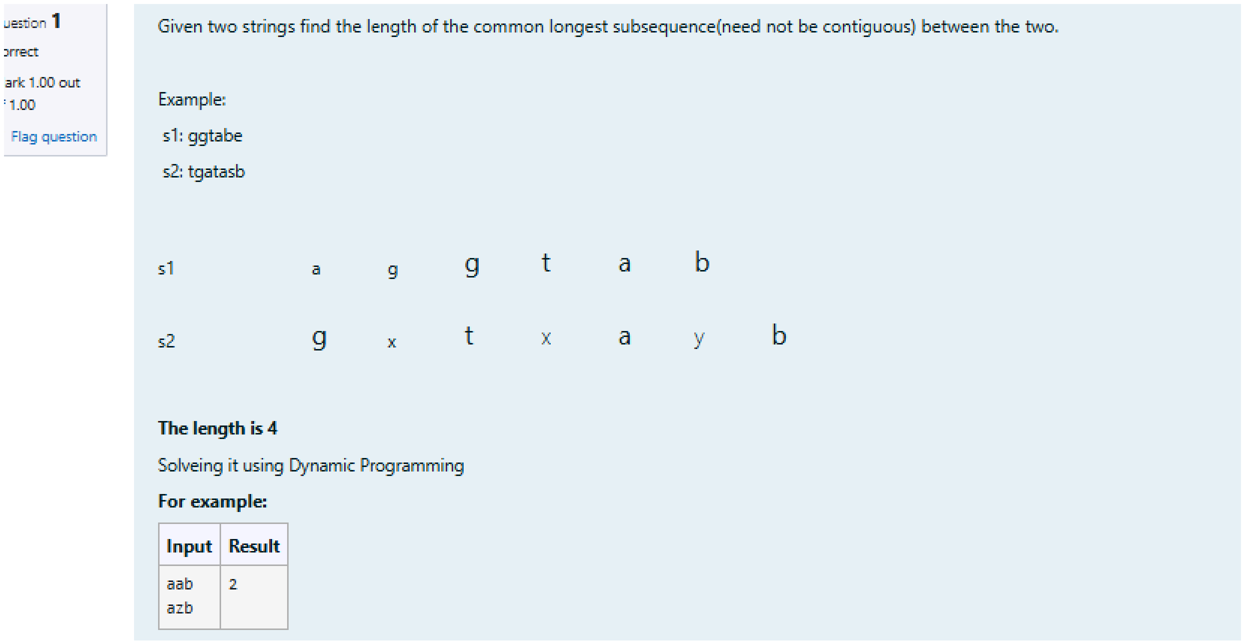


	Input	Expected	Got	
✓	3 1 2 4 2 3 4 8 7 1	19	19	✓
✓	3 1 3 1 1 5 1 4 2 1	12	12	✓
✓	4 1 1 3 4 1 5 7 8 2 3 4 6 1 6 9 0	28	28	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.



uestion **1**

orrect

ark 1.00 out

f 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

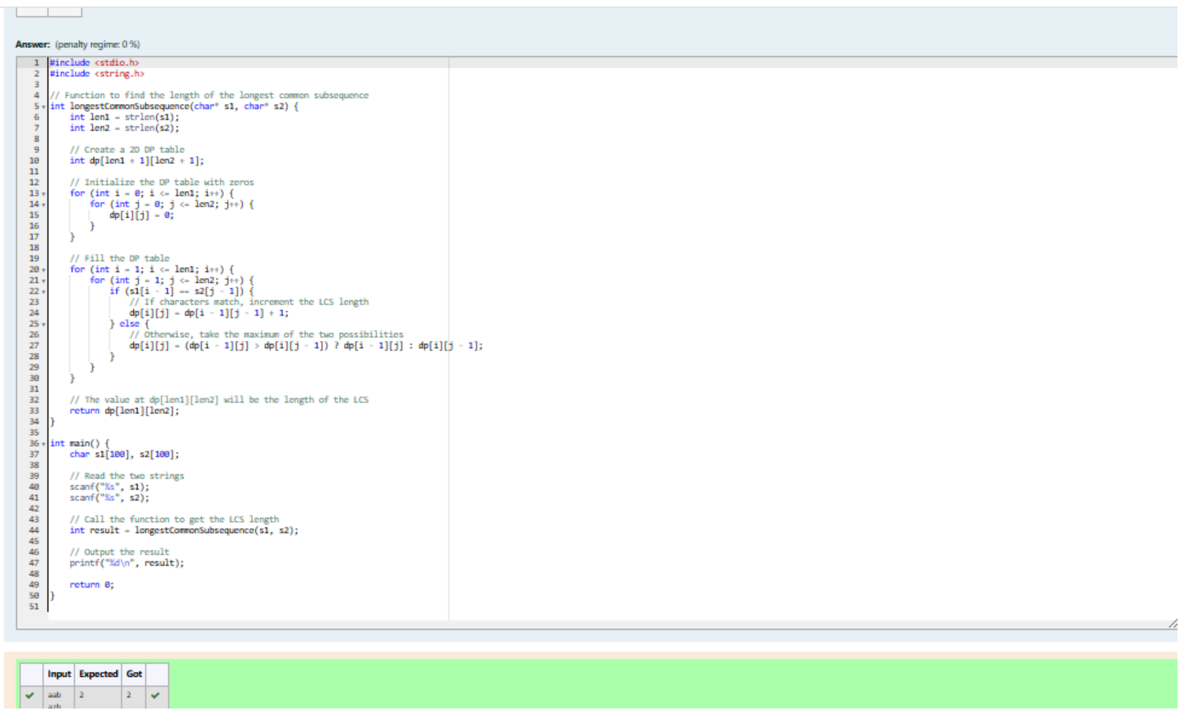
s1	a	g	g	t	a	b	
s2	g	x	t	x	a	y	b

The length is 4

Solveing it using Dynamic Programming

For example:

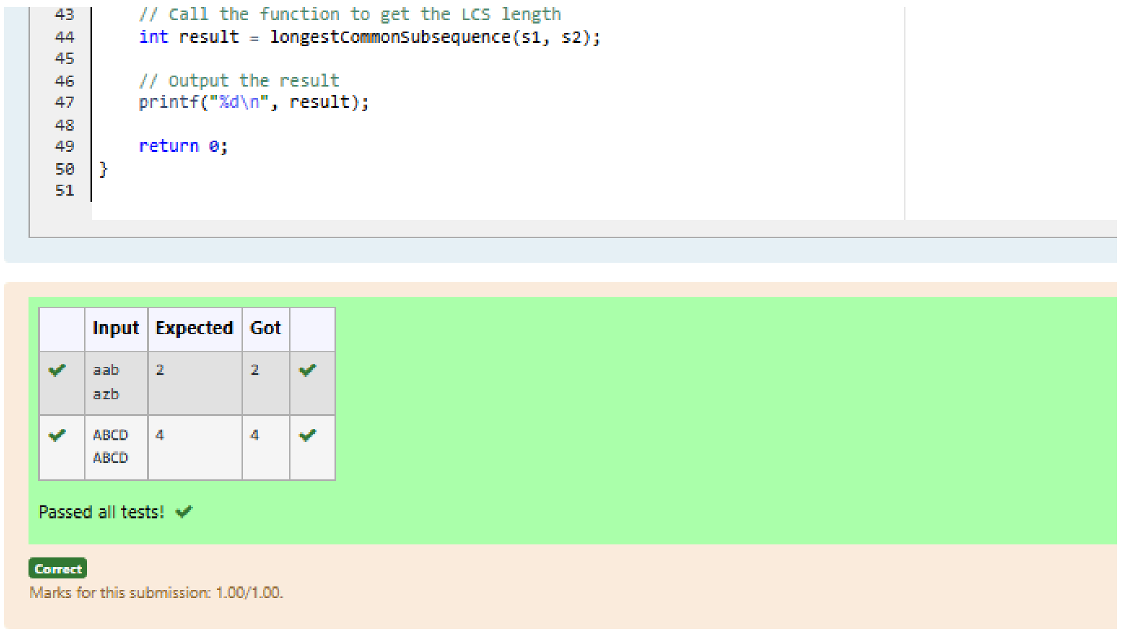
Input	Result
aab azb	2



Answer: (penalty regime: 0 %)

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

// Function to find the length of the longest common subsequence
int longestCommonSubsequence(char* s1, char* s2) {
    int len1 = strlen(s1);
    int len2 = strlen(s2);

    // Create a 2D DP table
    int dp[len1 + 1][len2 + 1];

    // Initialize the DP table with zeros
    for (int i = 0; i <= len1; i++) {
        for (int j = 0; j <= len2; j++) {
            dp[i][j] = 0;
        }
    }

    // Fill the DP table
    for (int i = 1; i <= len1; i++) {
        for (int j = 1; j <= len2; j++) {
            if (s1[i - 1] == s2[j - 1]) {
                // If characters match, increment the LCS length
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                // Otherwise, take the maximum of the two possibilities
                dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : dp[i][j - 1];
            }
        }
    }

    // The value at dp[len1][len2] will be the length of the LCS
    return dp[len1][len2];
}

int main() {
    char s1[100], s2[100];

    // Read the two strings
    scanf("%s", s1);
    scanf("%s", s2);

    // Call the function to get the LCS length
    int result = longestCommonSubsequence(s1, s2);

    // Output the result
    printf("%d\n", result);

    return 0;
}
```

	Input	Expected	Got	
✓	aab	2	2	✓



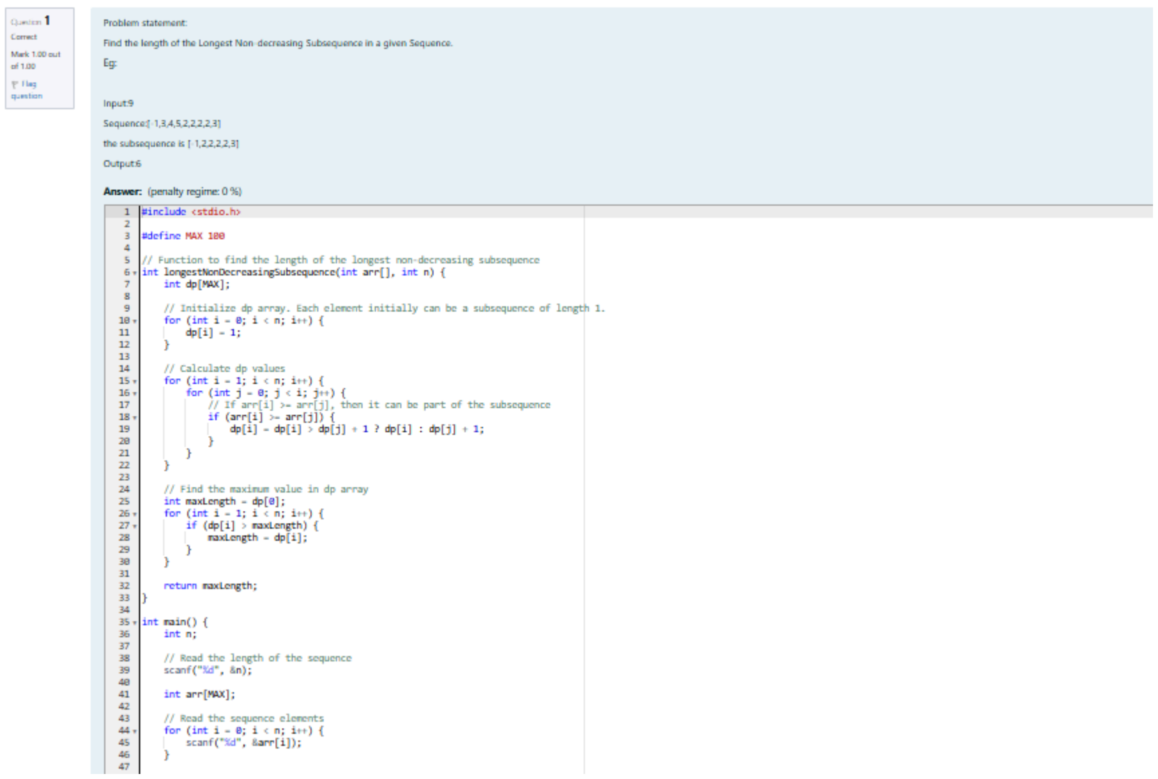
```c
    // Call the function to get the LCS length
    int result = longestCommonSubsequence(s1, s2);

    // Output the result
    printf("%d\n", result);

    return 0;
}
```

	Input	Expected	Got	
✓	aab azb	2	2	✓
✓	ABCD ABCD	4	4	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



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

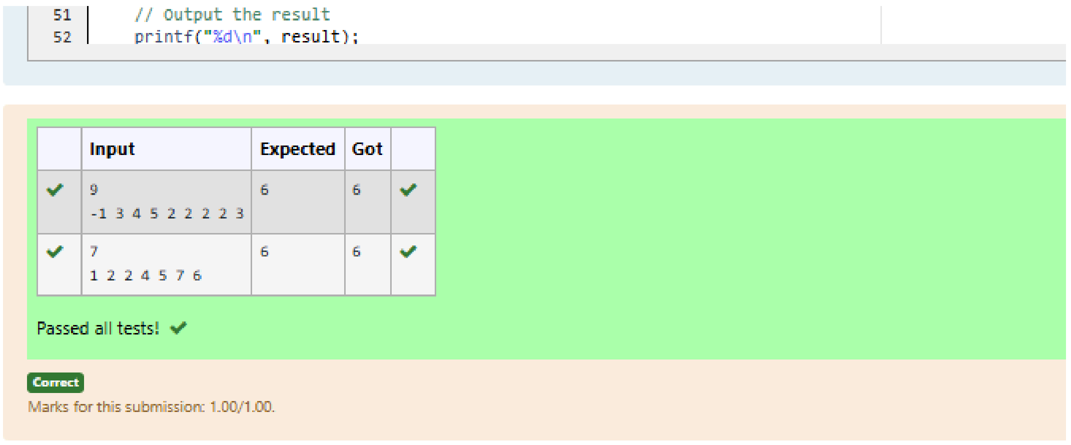
Sequence:[-1,3,4,5,2,2,2,2,3]

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

Output:6

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

#define MAX 100

// Function to find the length of the longest non-decreasing subsequence
int longestNonDecreasingSubsequence(int arr[], int n) {
    int dp[MAX];

    // Initialize dp array. Each element initially can be a subsequence of length 1.
    for (int i = 0; i < n; i++) {
        dp[i] = 1;
    }

    // Calculate dp values
    for (int i = 1; i < n; i++) {
        for (int j = 0; j < i; j++) {
            // If arr[i] >= arr[j], then it can be part of the subsequence
            if (arr[i] >= arr[j]) {
                dp[i] = dp[i] > dp[j] + 1 ? dp[i] : dp[j] + 1;
            }
        }
    }

    // Find the maximum value in dp array
    int maxLength = dp[0];
    for (int i = 1; i < n; i++) {
        if (dp[i] > maxLength) {
            maxLength = dp[i];
        }
    }

    return maxLength;
}

int main() {
    int n;

    // Read the length of the sequence
    scanf("%d", &n);

    int arr[MAX];

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

```



```c
    // Output the result
    printf("%d\n", result);
```

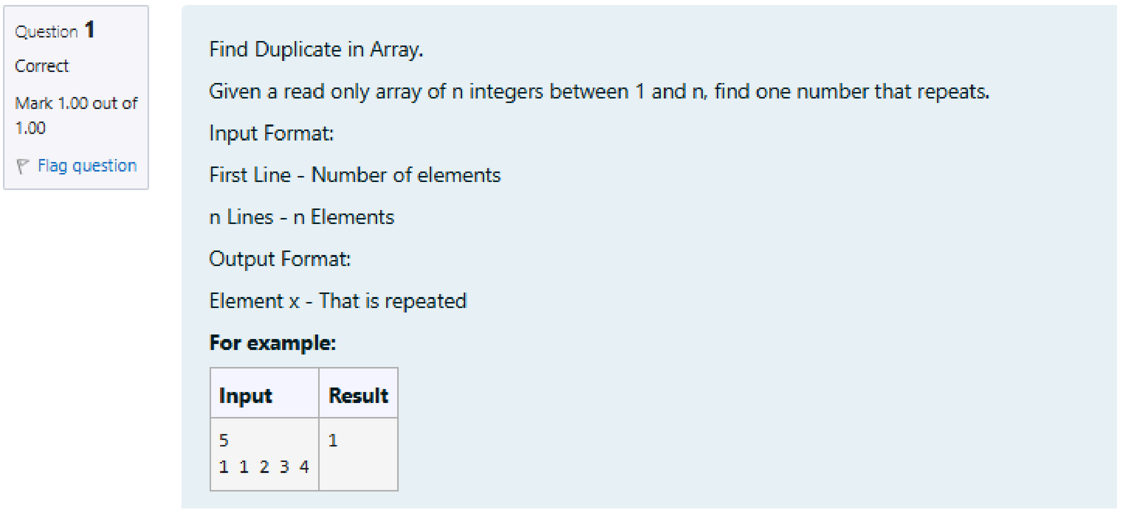
	Input	Expected	Got	
✓	9 -1 3 4 5 2 2 2 2 3	6	6	✓
✓	7 1 2 2 4 5 7 6	6	6	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

PROGRAMS ON COMPETITIVE PROGRAMMING



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

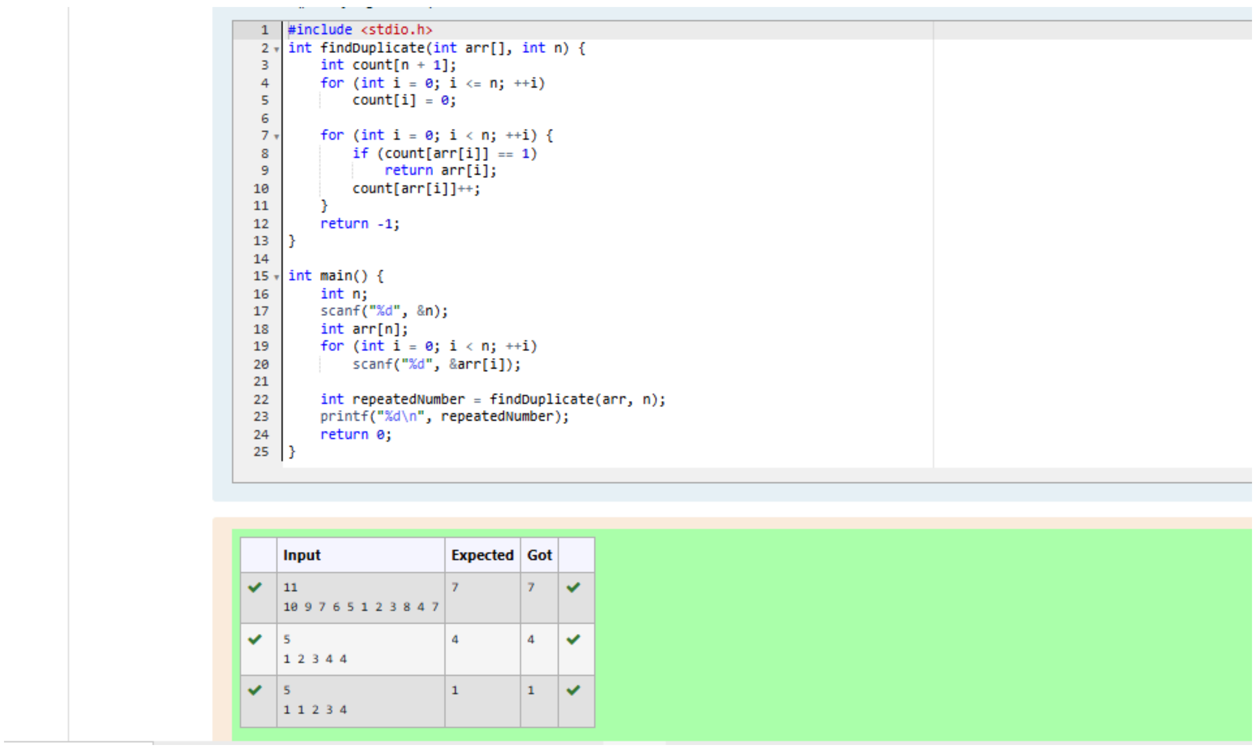
n Lines - n Elements

Output Format:

Element x - That is repeated

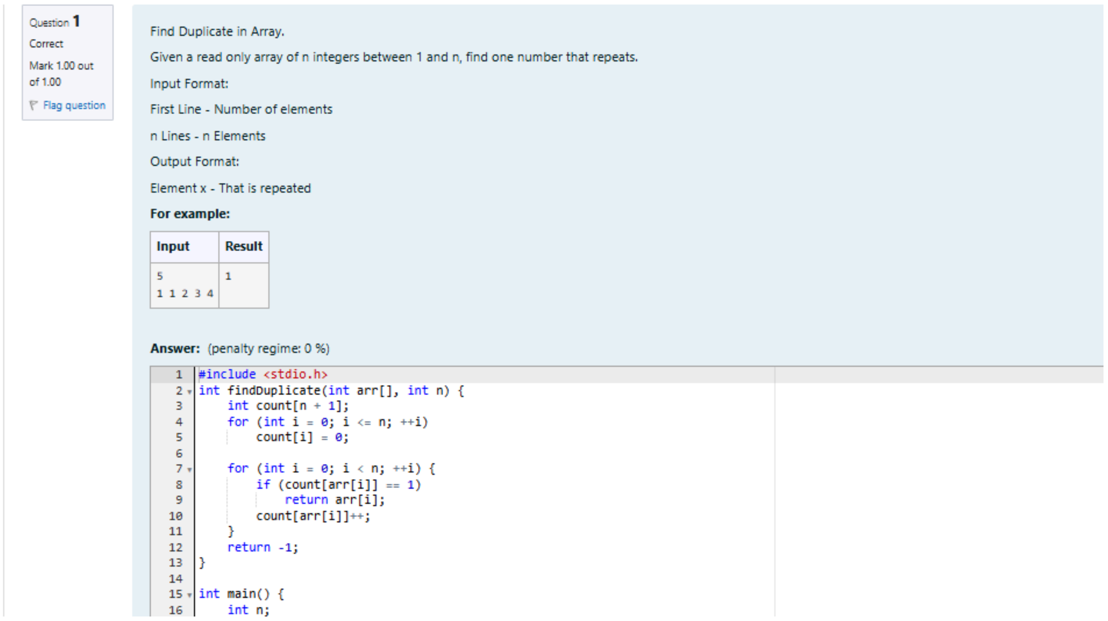
For example:

Input	Result
5 1 1 2 3 4	1



```c
#include <stdio.h>
int findDuplicate(int arr[], int n) {
    int count[n + 1];
    for (int i = 0; i <= n; ++i)
        count[i] = 0;

    for (int i = 0; i < n; ++i) {
        if (count[arr[i]] == 1)
            return arr[i];
        count[arr[i]]++;
    }
    return -1;
}

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

    int repeatedNumber = findDuplicate(arr, n);
    printf("%d\n", repeatedNumber);
    return 0;
}
```

	Input	Expected	Got	
✓	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✓
✓	5 1 2 3 4 4	4	4	✓
✓	5 1 1 2 3 4	1	1	✓



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

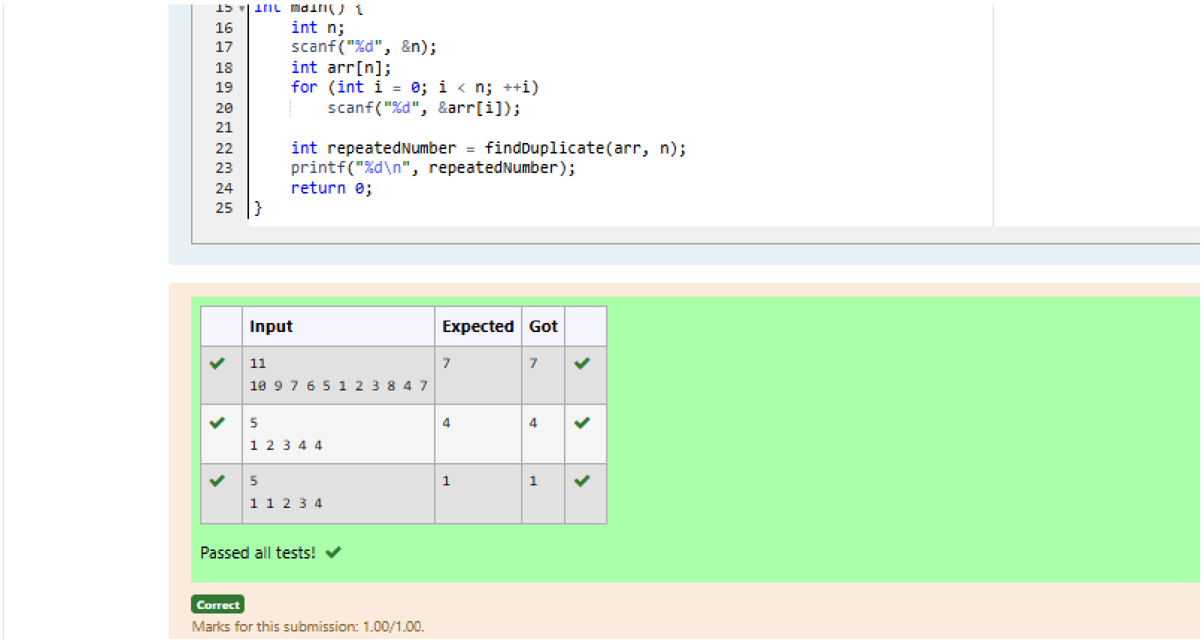
Output Format:

Element x - That is repeated

For example:

Input	Result
5 1 1 2 3 4	1

Answer: (penalty regime: 0 %)



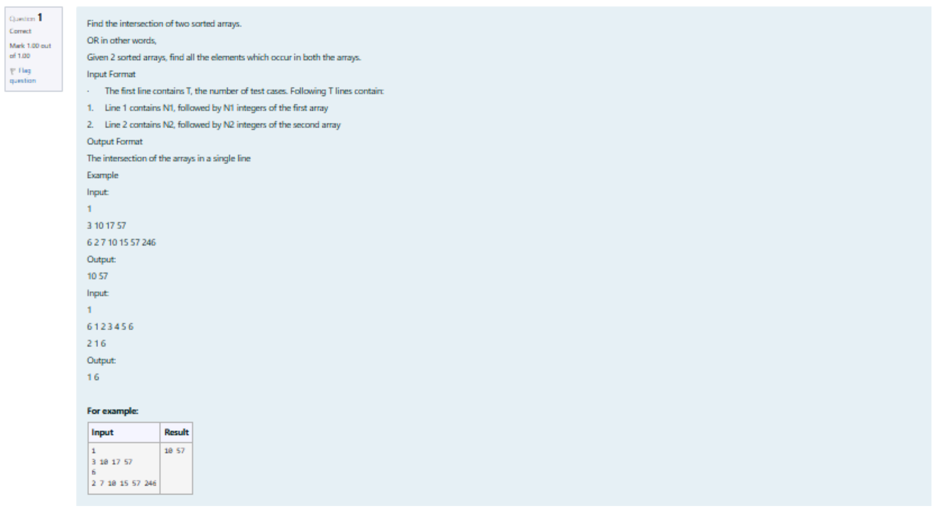
```
#include <stdio.h>
int findDuplicate(int arr[], int n) {
    int count[n + 1];
    for (int i = 0; i <= n; ++i)
        count[i] = 0;

    for (int i = 0; i < n; ++i) {
        if (count[arr[i]] == 1)
            return arr[i];
        count[arr[i]]++;
    }
    return -1;
}

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

    int repeatedNumber = findDuplicate(arr, n);
    printf("%d\n", repeatedNumber);
    return 0;
}
```

	Input	Expected	Got	
✓	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✓
✓	5 1 2 3 4 4	4	4	✓
✓	5 1 1 2 3 4	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

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

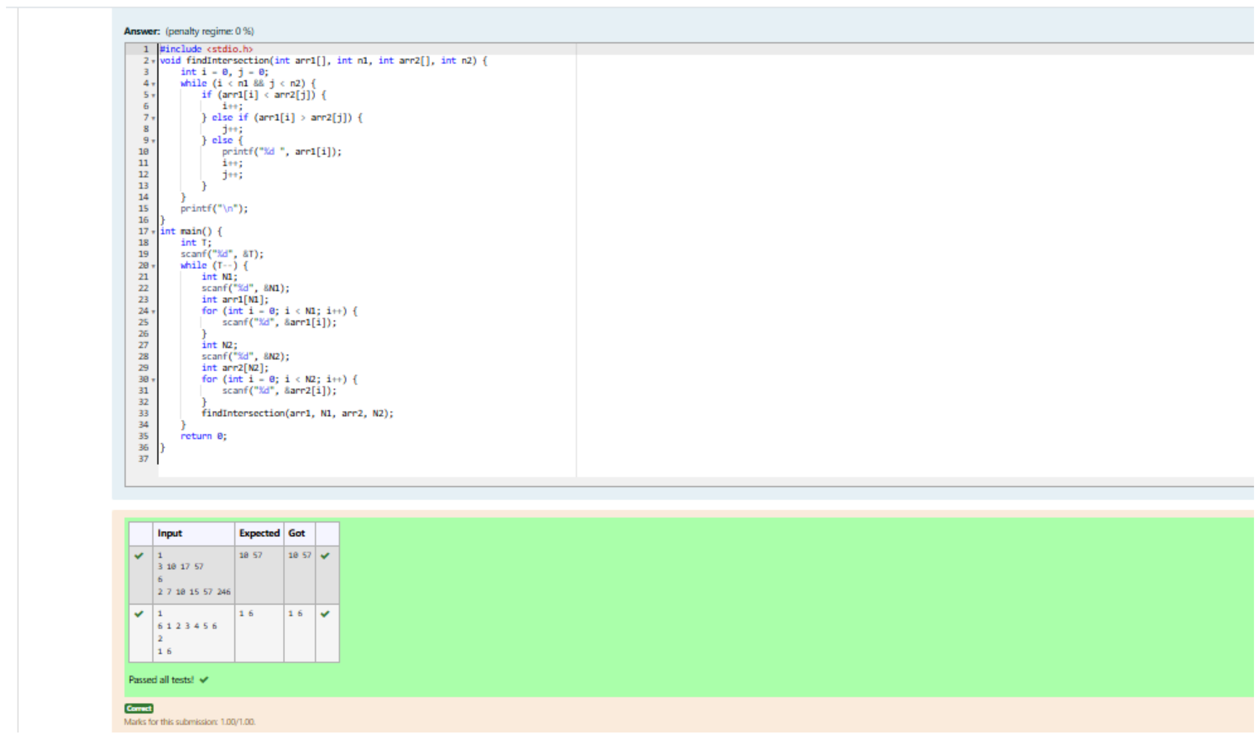
Output:

1 6

For example:

Input	Result

1
3 10 17 57
6
2 7 10 15 57 246
	10 57



Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
void findIntersection(int arr1[], int n1, int arr2[], int n2) {
    int i = 0, j = 0;
    while (i < n1 && j < n2) {
        if (arr1[i] < arr2[j]) {
            i++;
        } else if (arr1[i] > arr2[j]) {
            j++;
        } else {
            printf("%d ", arr1[i]);
            i++;
            j++;
        }
    }
    printf("\n");
}
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N1;
        scanf("%d", &N1);
        int arr1[N1];
        for (int i = 0; i < N1; i++) {
            scanf("%d", &arr1[i]);
        }
        int N2;
        scanf("%d", &N2);
        int arr2[N2];
        for (int i = 0; i < N2; i++) {
            scanf("%d", &arr2[i]);
        }
        findIntersection(arr1, N1, arr2, N2);
    }
    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.

Correct

Mark 1.00 out of 1.00

Flag question

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
3 10 17 57
6
2 7 10 15 57 246
	10 57

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
void findIntersection(int arr1[], int n1, int arr2[], int n2) {
    int i = 0, j = 0;
    int found = 0;
    while (i < n1 && j < n2) {
        if (arr1[i] < arr2[j]) {
            i++;
        } else if (arr1[i] > arr2[j]) {
            j++;
        } else {
            if (found == 0) {
                printf("%d", arr1[i]);
                found = 1;
            } else {
                printf(" %d", arr1[i]);
            }
            i++;
            j++;
        }
    }
    if (found == 0) {
        printf("No intersection");
}
}
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int n1, n2;
        scanf("%d", &n1);
        int arr1[n1];
        for (int i = 0; i < n1; i++) {
            scanf("%d", &arr1[i]);
        }
        scanf("%d", &n2);
        int arr2[n2];
        for (int i = 0; i < n2; i++) {
            scanf("%d", &arr2[i]);
        }
        findIntersection(arr1, n1, arr2, n2);
        printf("\n");
    }
    return 0;
}

```

```
        }
        findIntersection(arr1, n1, arr2, n2);
        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.

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:

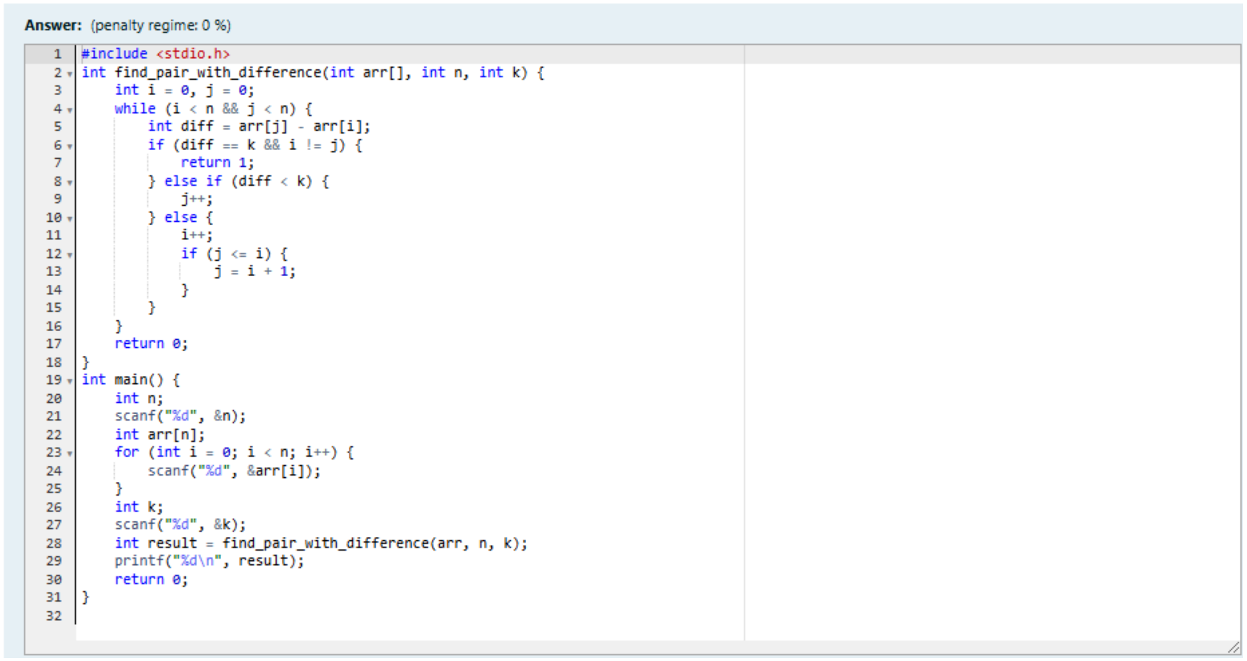
YES as 5 - 1 = 4

So Return 1.

For example:

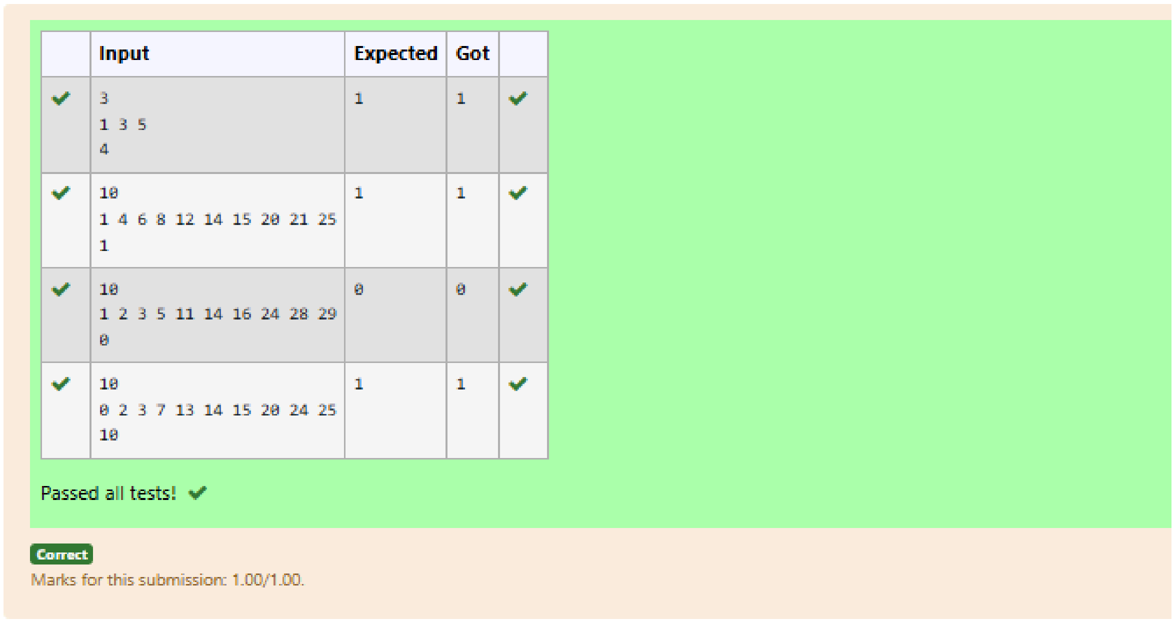
Input	Result

3
1 3 5
4
	1



Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int find_pair_with_difference(int arr[], int n, int k) {
    int i = 0, j = 0;
    while (i < n && j < n) {
        int diff = arr[j] - arr[i];
        if (diff == k && i != j) {
            return 1;
        } else if (diff < k) {
            j++;
        } else {
            i++;
            if (j <= i) {
                j = i + 1;
            }
        }
    }
    return 0;
}
int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int k;
    scanf("%d", &k);
    int result = find_pair_with_difference(arr, n, k);
    printf("%d\n", result);
    return 0;
}

```

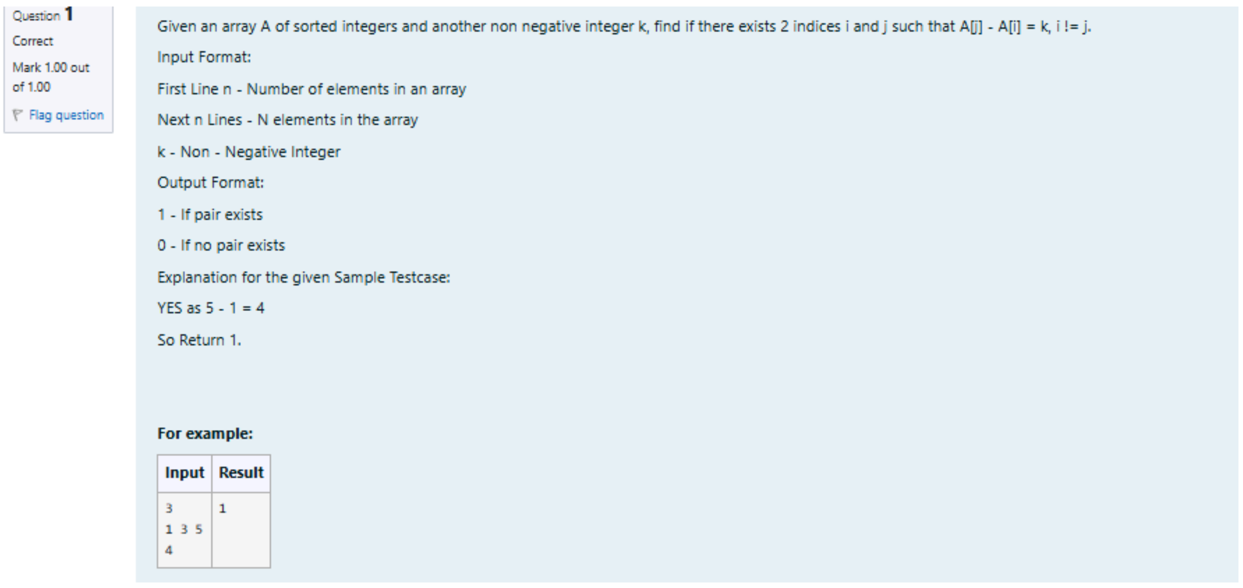


	Input	Expected	Got	
✓	3 1 3 5 4	1	1	✓
✓	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✓
✓	10 1 2 3 5 11 14 16 24 28 29 0	0	0	✓
✓	10 0 2 3 7 13 14 15 20 24 25 10	1	1	✓

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 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:

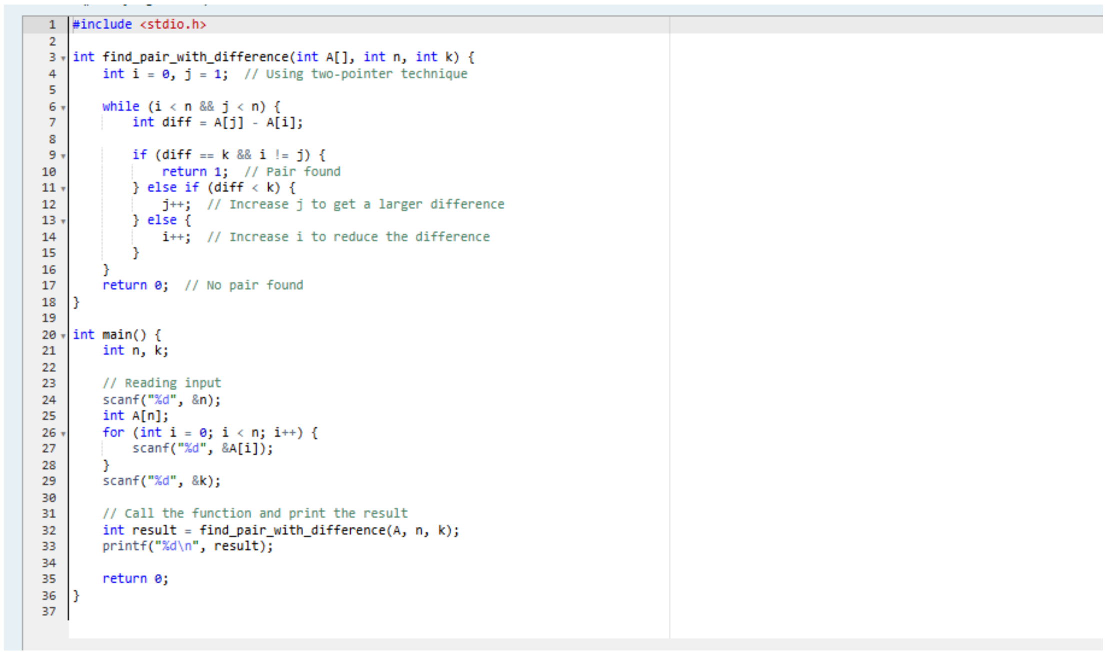
YES as 5 - 1 = 4

So Return 1.

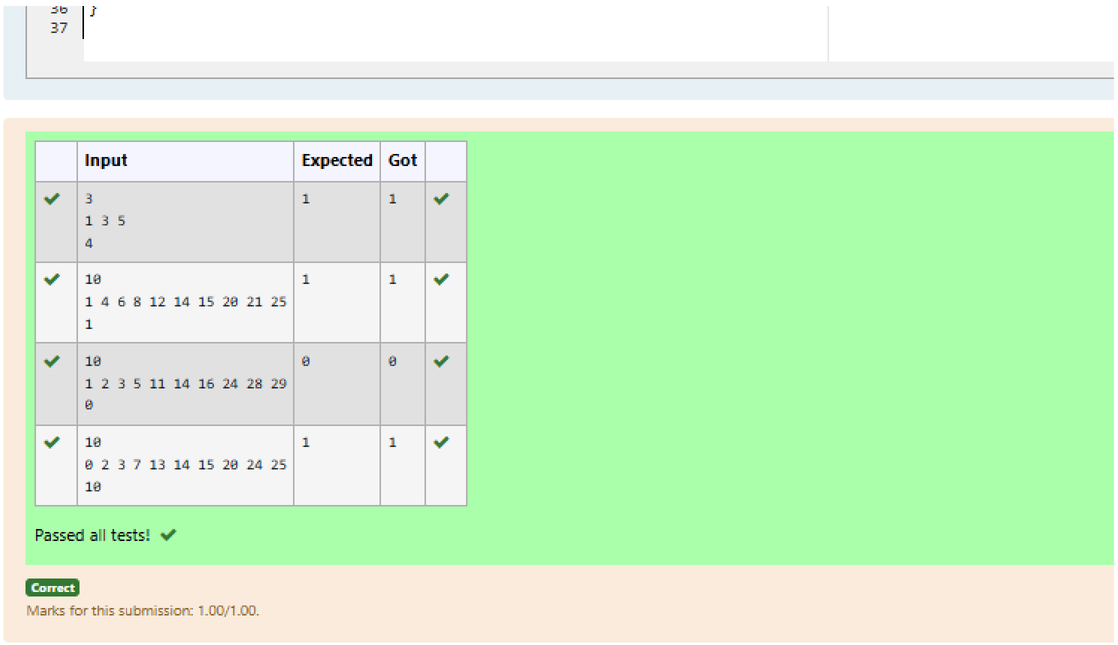
For example:

Input	Result

3
1 3 5
4
	1



```c
#include <stdio.h>

int find_pair_with_difference(int A[], int n, int k) {
    int i = 0, j = 1;  // Using two-pointer technique

    while (i < n && j < n) {
        int diff = A[j] - A[i];

        if (diff == k && i != j) {
            return 1;  // Pair found
        } else if (diff < k) {
            j++;  // Increase j to get a larger difference
        } else {
            i++;  // Increase i to reduce the difference
        }
    }
    return 0;  // No pair found
}

int main() {
    int n, k;

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

    // Call the function and print the result
    int result = find_pair_with_difference(A, n, k);
    printf("%d\n", result);

    return 0;
}
```



```
}
```

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

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.