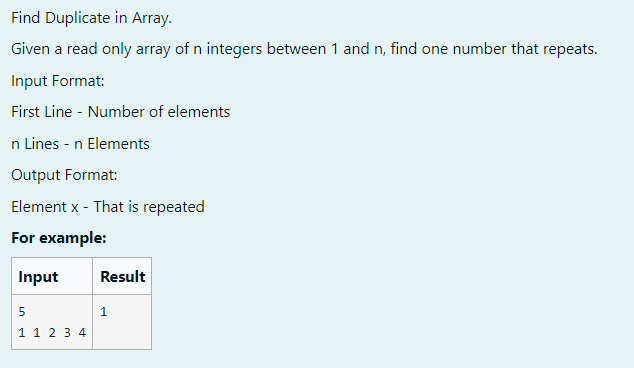
COMPETITIVE PROGRAMMING

**QUESTION 6.A AIM:**



**ALGORITHM:**

**Step 1: Start**

**Step 2:** Input the integer n, the number of elements in the array.

**Step 3:** Input n integers into an array a.

**Step 4:** Initialize r as -1 to store the repeated element.

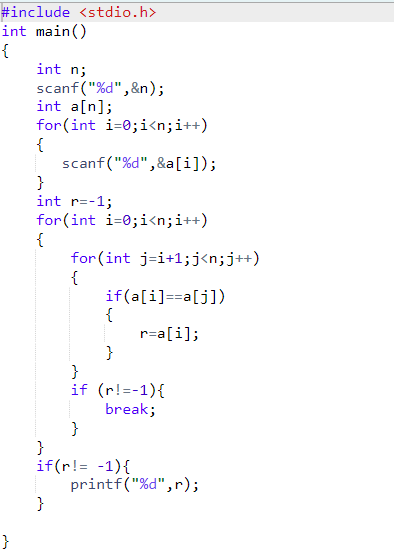
**Step 5:** Use a nested loop to check if any element a[i] matches with subsequent elements a[j].

**Step 6:** If a match is found, set r to the repeated element.

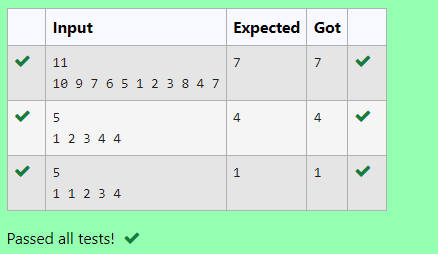
**Step 7:** If a repeated element is found (r != -1), print the repeated element.

**Step 8: Stop**

**PROGRAM**:



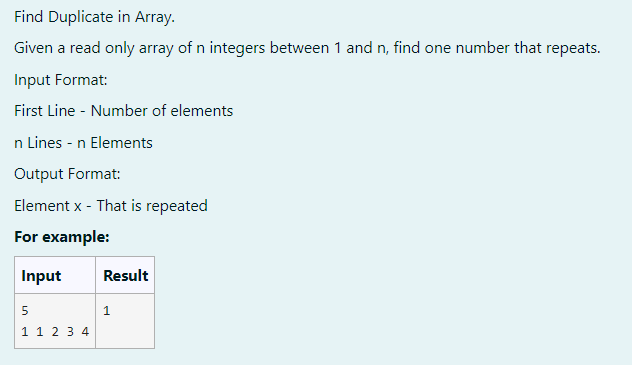
**OUTPUT**:



**RESULT:**

The above progeam is executed successfully .

**QUESTION 6.B AIM:**



**ALGORITHM:**

**Step 1: Start**

**Step 2:** Input the integer n, the number of elements in the array.

**Step 3:** Input n integers into an array a.

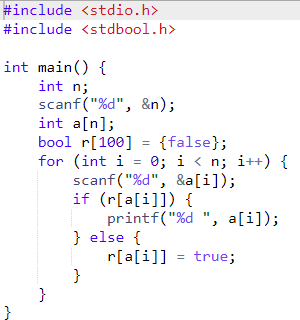
**Step 4:** Initialize a boolean array r[100] to track whether a number has already been encountered.

**Step 5:** Iterate through the array a. For each element, check if it has already been seen.

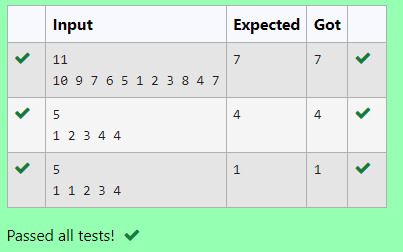
**Step 6:** If the element is already seen, print it. If not, mark it as seen in r.

**Step 7: Stop**

**PROGRAM**:



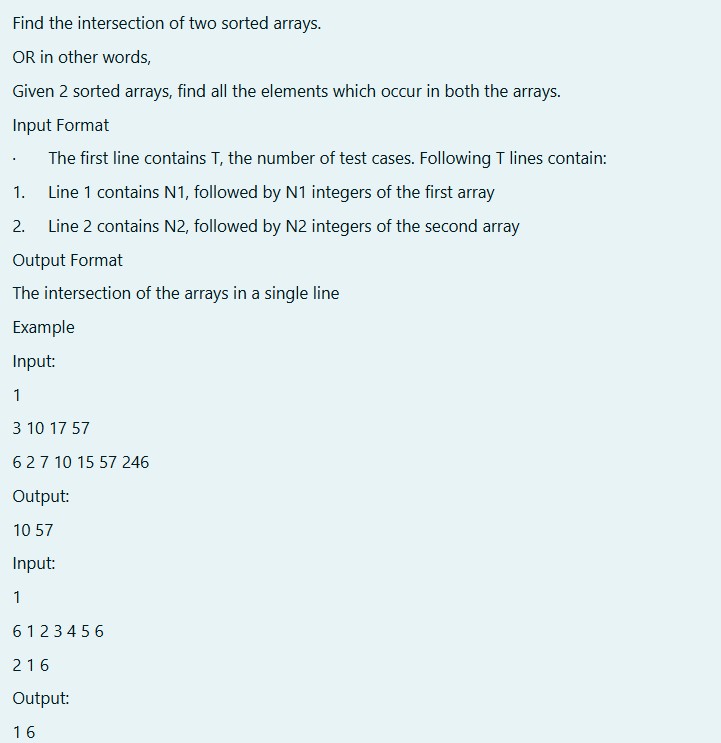
**OUTPUT:**



**RESULT:**

The above program is executed successfully.

**Question 6.C AIM:**



**ALLGORITHM**:

**Step 1: Start**

**Step 2:** Input the number of test cases t.

**Step 3:** For each test case, input the size n1 of the first array and input the array arr1.

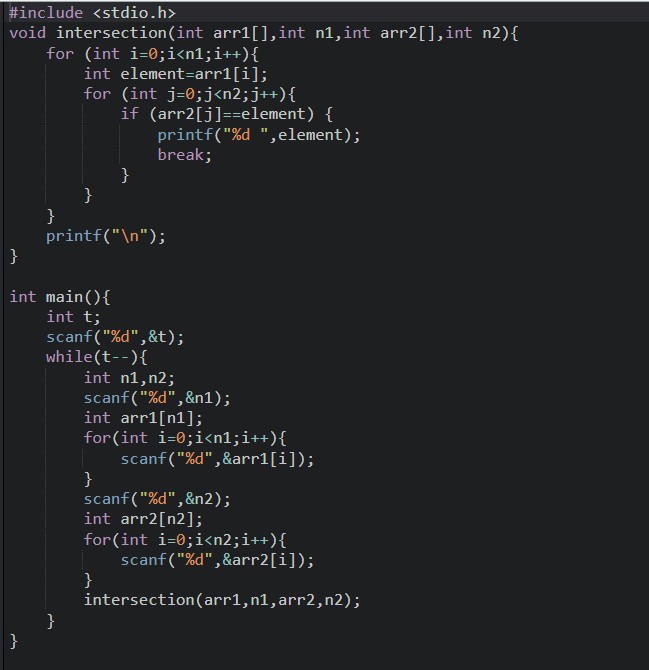
**Step 4:** Input the size n2 of the second array and input the array arr2.

**Step 5:** For each element of arr1, check if it exists in arr2.

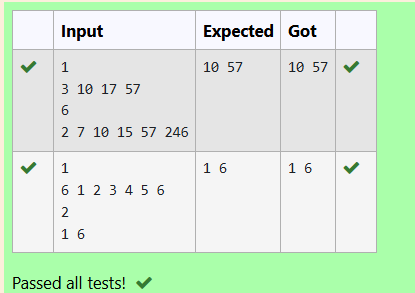
**Step 6:** If a match is found, print the element as part of the intersection.

**Step 7: Stop**

**PROGRAM:**



**OUTPUT:**

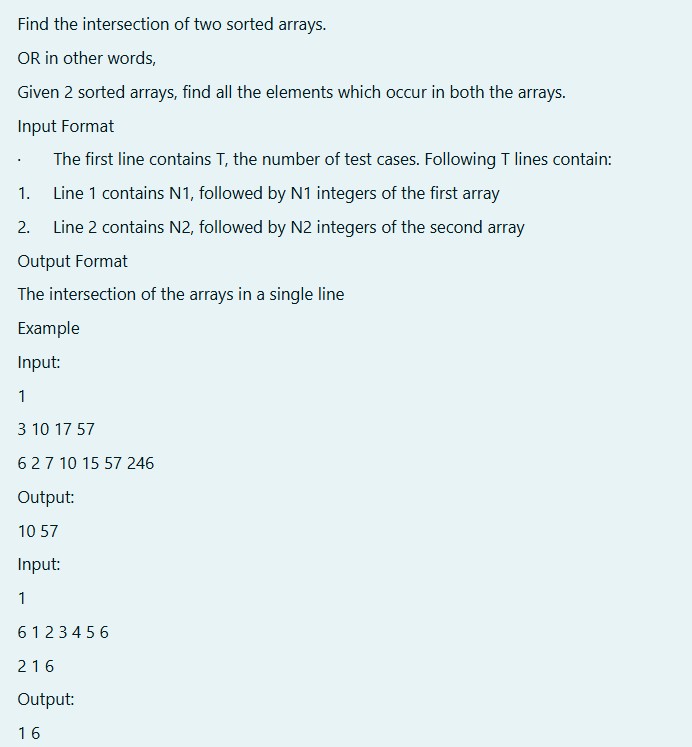


**RESULT:**

The above program is executed successfully.

**Question 6.D**

AIM:



**ALGORITM:**

**Step 1: Start**

**Step 2:** Input the number of test cases t.

**Step 3:** For each test case, input the size n1 of the first array and input the array arr1.

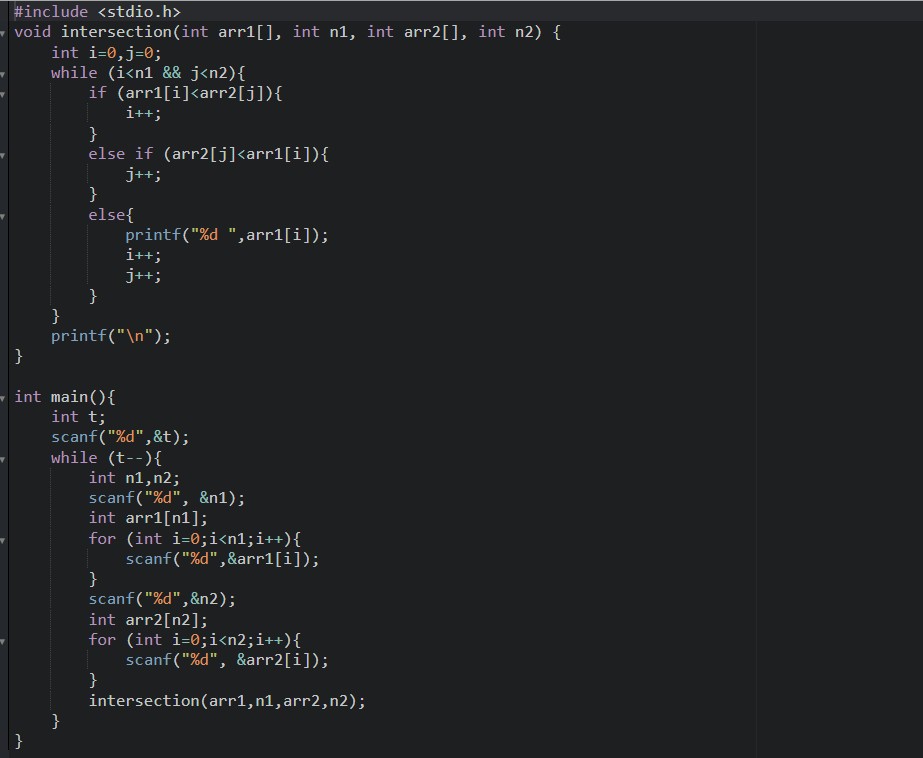
**Step 4:** Input the size n2 of the second array and input the array arr2.

**Step 5:** Initialize two indices i and j to 0 and use them to traverse both arrays.

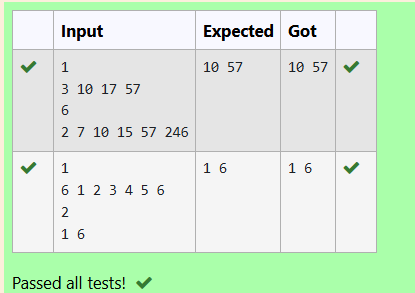
**Step 6:** If arr1[i] < arr2[j], increment i. If arr2[j] < arr1[i], increment j. **Step 7:** If arr1[i] == arr2[j], print the common element and increment both i and j. **Step 8:** Continue until one of the arrays is completely traversed.

**Step 9: Stop**

**PROGRAM:**



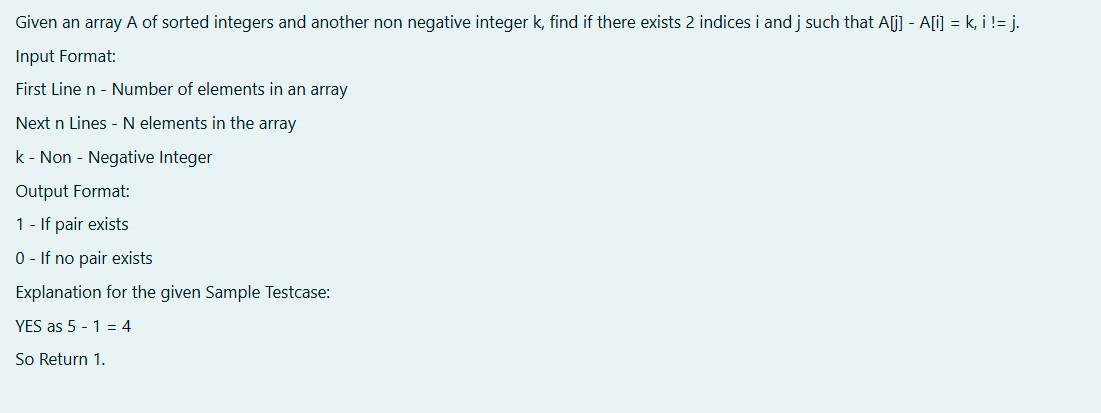
**OUTPUT:**



**RESULT:**

The above program is executed successfully.

**Question 6.E AIM:**



**ALGORITHM**:

**Step 1: Start**

**Step 2:** Input the integer n (number of elements) and the array arr.

**Step 3:** Input the integer k (difference to check for).

**Step 4:** Use a nested loop to compare each pair of elements arr[i] and arr[j].

**Step 5:** If the difference arr[j] - arr[i] == k, return 1.

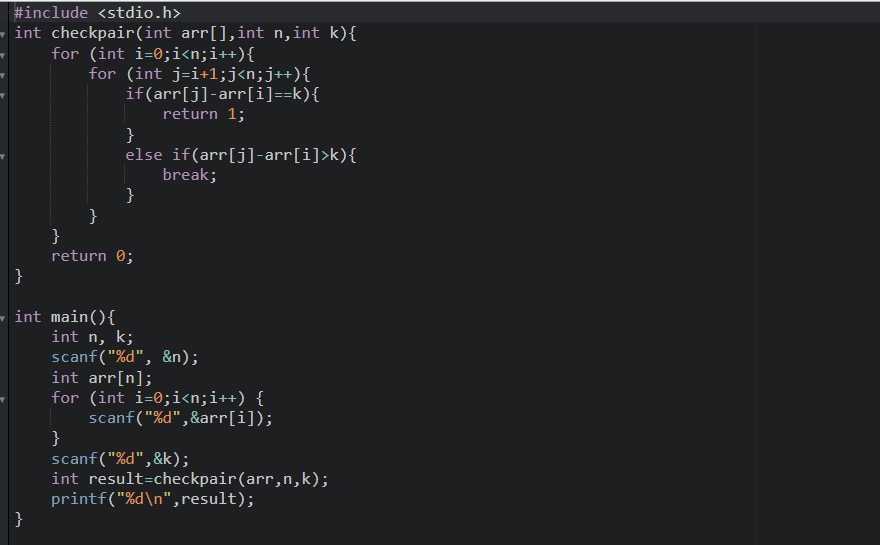
**Step 6:** If the difference exceeds k, break the inner loop.

**Step 7:** If no valid pair is found, return 0.

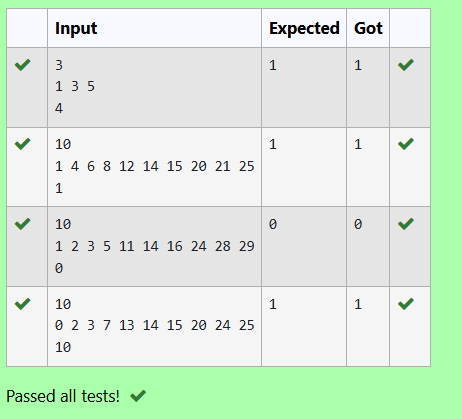
**Step 8:** Output the result.

**Step 9: Stop**

**PROGRAM :**



**OUTPUT :**

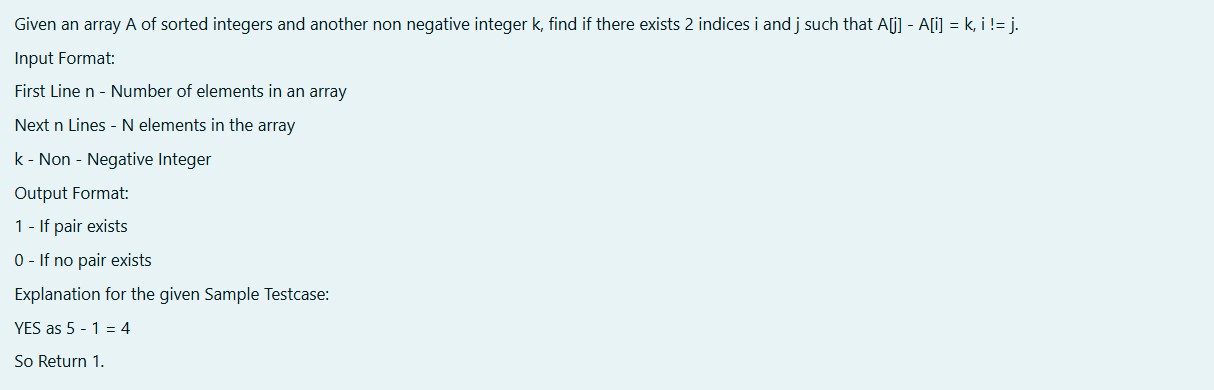


**RESULT:**

The above program is executed successfully.

**Question 6.F**

**AIM:**



**ALGORITHM:**

**Step 1: Start**

**Step 2: Input the integer n (number of elements) and the array arr. Step 3: Input the integer k (difference to check for).**

**Step 4: Initialize two indices i = 0 and j = 1.**

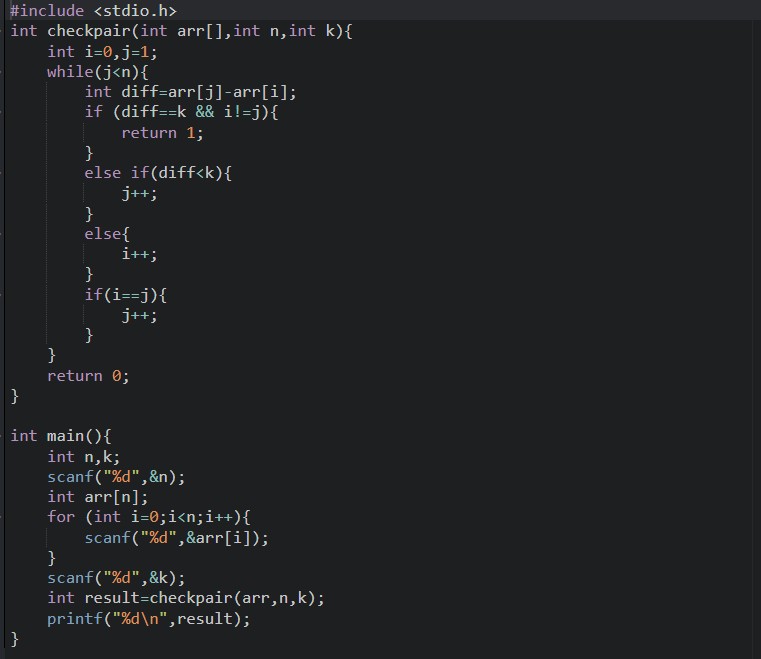
**Step 5: While j < n, calculate the difference arr[j] - arr[i]. Step 6: If the difference is k, return 1.**

**Step 7: If the difference is less than k, increment j. If the difference is greater, increment i. Step 8: If i == j, increment j to avoid comparing the same element with itself.**

**Step 9: If no valid pair is found, return 0. Step 10: Output the result.**

**Step 11: Stop**

**PROGRAM:**



**OUTPUT :**



**RESULT :**

**The above program is executed successfully.**