**[COMPETITIVE PROGRAMMING](http://118.185.187.137/moodle/course/view.php?id=158" \l "section-1)**

**AIM:**

To implement an algorithm to find one duplicate number in a read-only array of integers between 1 and n with a time complexity of O(n²) and constant space complexity.

**PROGRAM:**

#include <stdio.h>

int fd(int arr[], int n) {

int slow = arr[0];

int fast = arr[arr[0]];

while (slow != fast) {

slow = arr[slow];

fast = arr[arr[fast]];

}

slow = 0;

while (slow != fast) {

slow = arr[slow];

fast = arr[fast];

}

return slow;

}

int main() {

int n;

scanf("%d", &n);

int arr[n];

for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

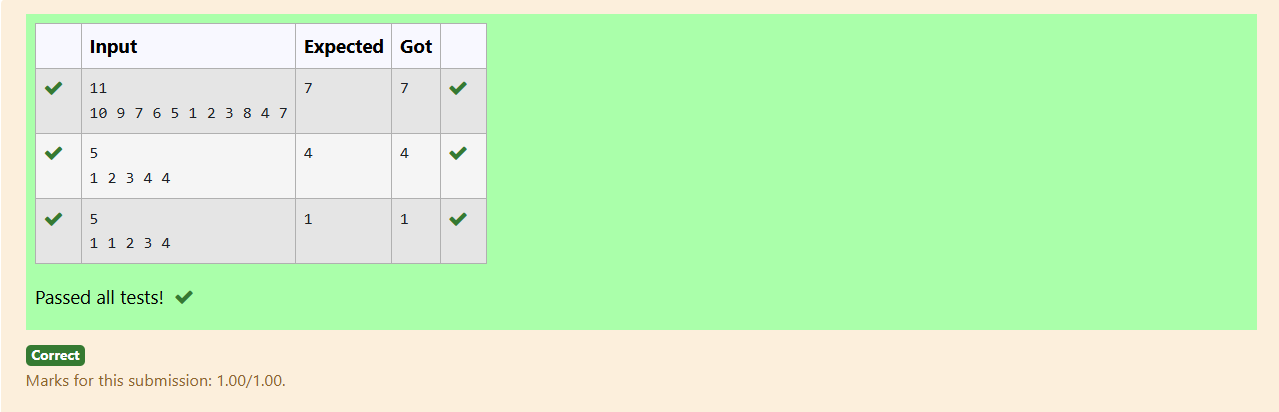
}

int dup=fd(arr, n);

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

}

**OUTPUT:**



**AIM:**

To implement an efficient algorithm to find one duplicate number in a read-only array of integers between 1 and n with a time complexity of O(n) and constant space complexity.  
**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

int findDuplicate(int arr[], int n) {

for (int i = 0; i < n; i++) {

int index = abs(arr[i]) - 1;

if (arr[index] < 0) {

return abs(arr[i]);

}

arr[index] = -arr[index];

}

return -1;

}

int main() {

int n;

scanf("%d", &n);

int arr[n];

for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

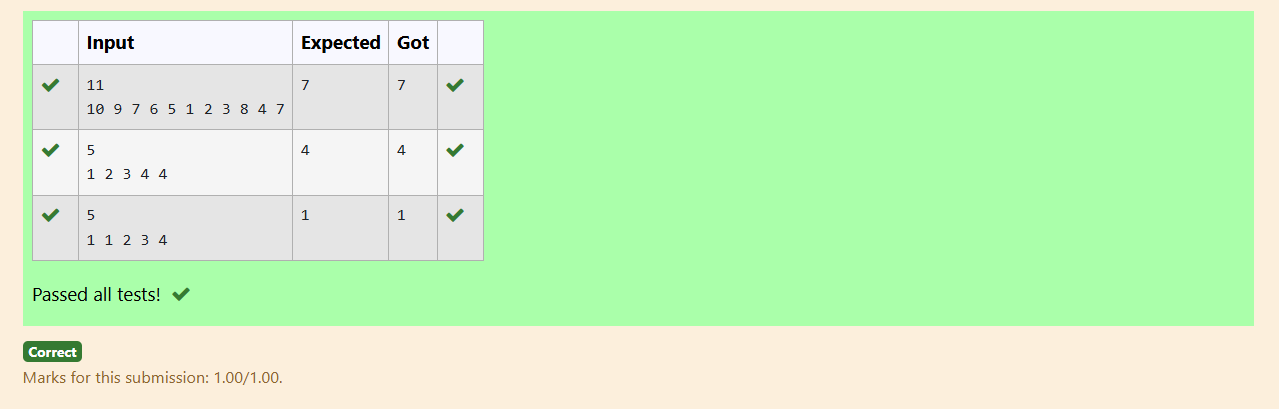
int duplicate = findDuplicate(arr, n);

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

return 0;

}

**OUTPUT:**



**AIM:**

To implement an algorithm that finds the intersection of two sorted arrays, returning elements common to both arrays with a time complexity of O(m\*n) and constant space complexity.

**PROGRAM:**

#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]) {

printf("%d ", arr1[i]);

i++;

j++;

}

else if (arr1[i] < arr2[j]) {

i++;

} else {

j++;

}

}

printf("\n");

}

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]);

}

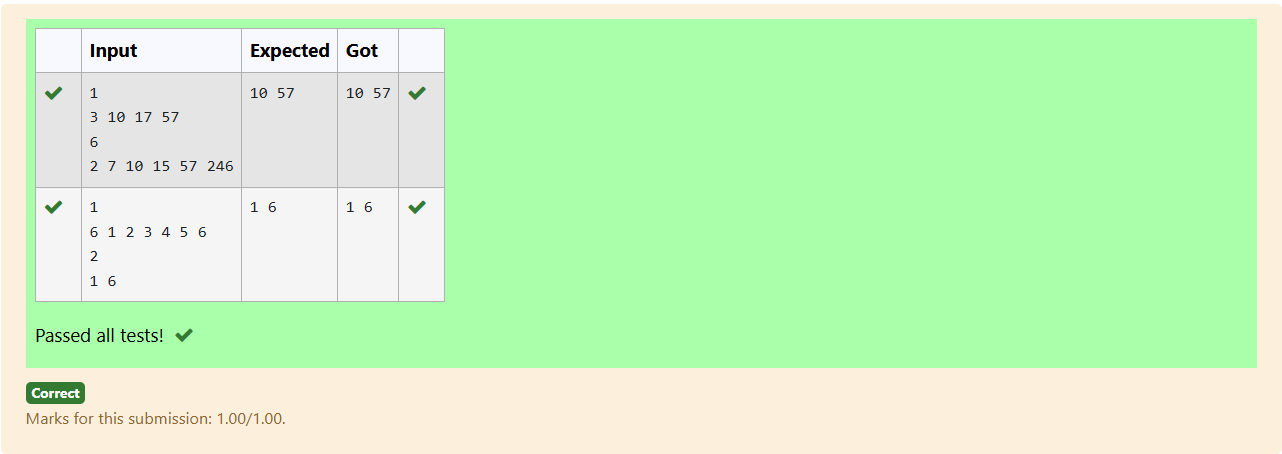
findIntersection(arr1, n1, arr2, n2);

}

return 0;

}

**OUTPUT:**



**AIM:**

To implement an efficient algorithm that finds the intersection of two sorted arrays, returning elements common to both arrays with a time complexity of O(m + n) and constant space complexity.

**PROGRAM:**

#include <stdio.h>

void findIntersection(int arr1[], int n1, int arr2[], int n2) {

int i = 0, j = 0;

while (i < n1 && j < n2) {

if (i > 0 && arr1[i] == arr1[i - 1]) {

i++;

continue;

}

if (arr1[i] == arr2[j]) {

printf("%d ", arr1[i]);

i++;

j++;

}

else if (arr1[i] < arr2[j]) {

i++;

} else {

j++;

}

}

printf("\n");

}

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]);

}

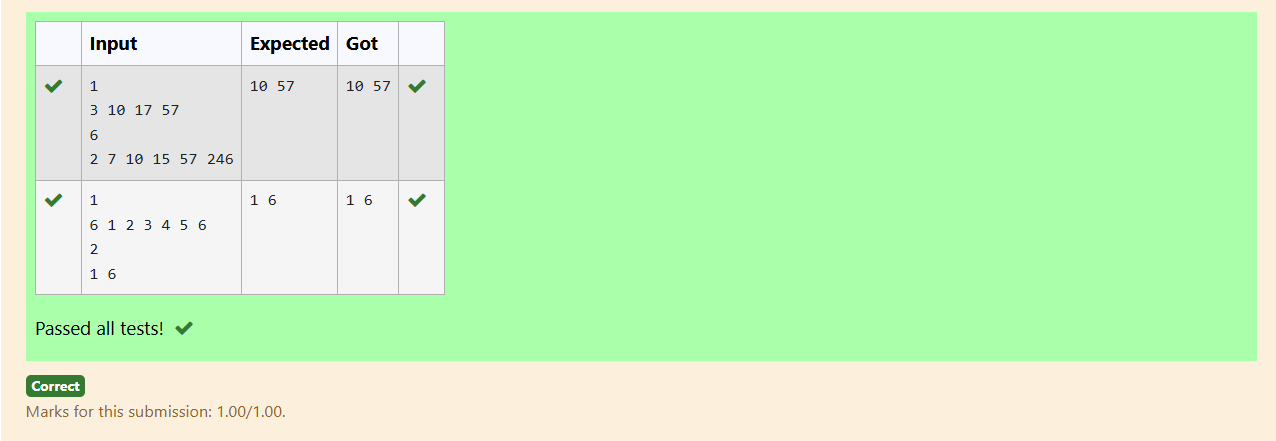
findIntersection(arr1, n1, arr2, n2);

}

return 0;

}

**OUTPUT:**



**AIM:**

To implement an algorithm to check if there exists a pair of indices i and j in a sorted array such that the difference between the elements at these indices equals a given integer k, with a time complexity of O(n²) and constant space complexity.

**PROGRAM:**

#include <stdio.h>

int findPairWithDifference(int arr[], int n, int k) {

for (int i = 0; i < n; i++) {

for (int j = i + 1; j < n; j++) {

if (arr[j] - arr[i] == k) {

return 1;

}

}

}

return 0;

}

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 result = findPairWithDifference(arr, n, k);

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

return 0;

}

**OUTPUT:**



**AIM:**

To implement an efficient algorithm to check if there exists a pair of indices i and j in a sorted array such that the difference between the elements at these indices equals a given integer k, with a time complexity of O(n) and constant space complexity.

**PROGRAM:**

#include <stdio.h>

int findPairWithDifference(int arr[], int n, int k) {

int i = 0, j = 1;

while (j < n) {

int diff = arr[j] - arr[i];

if (diff == k && i != j) {

return 1;

} else if (diff < k) {

j++;

} else {

i++;

}

if (i == j) {

j++;

}

}

return 0;

}

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 result = findPairWithDifference(arr, n, k);

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

}

**OUTPUT:**

