**COMPETITIVE PROGRAMMING**

**AIM:**

[1-Finding Duplicates-O(n^2) Time Complexity,O(1) Space Complexity](http://118.185.187.137/moodle/mod/quiz/view.php?id=1280)

**ALGORITHM:**

1. Read integer n (size of the array).
2. Create array arr of size n.
3. Fill arr with user input.
4. Loop through each index i from 0 to n-1:
   * Loop through each index j from i + 1 to n-1:
5. If no duplicates are found, the program will end without printing anything.

**PROBLEM:**

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 |

**PROGRAM:**

#include <stdio.h>

int main()

{

int n;

scanf("%d",&n);

int arr[n];

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

{

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

}

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

{

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

{

if (arr[i]==arr[j])

{

printf("%d",arr[j]);

return 0;

}

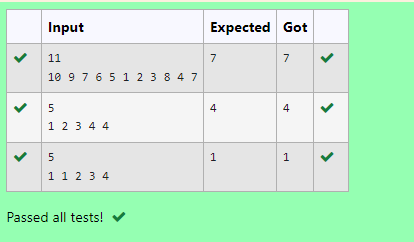
}

}

return 0;

}

**OUTPUT:**

****

**AIM:**

[2-Finding Duplicates-O(n) Time Complexity,O(1) Space Complexity](http://118.185.187.137/moodle/mod/quiz/view.php?id=1281)

**ALGORITHM:**

1. Read integer n (size of the array).
2. Create array arr of size n and initialize array s[100] to zero.
3. Loop through each index i from 0 to n-1:
   * Read integer input into arr[i].
   * If s[arr[i]] is non-zero (indicating the value has been seen before):Print
   * Set s[arr[i]] to n (marking the value as seen).
4. If no duplicates are found, the program will end without printing anything.

**PROBLEM:**

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 |

**PROGRAM:**

#include <stdio.h>

int main()

{

int n;

scanf("%d",&n);

int arr[n],s[100]={n};

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

{

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

if(s[arr[i]])

{

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

return 0;

}

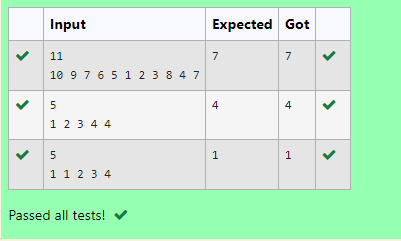
s[arr[i]]=n;

}

return 0;

}

**OUTPUT:**

****

**AIM:**

Find the intersection of two sorted arrays.

**ALGORITHM:**

1. Read integer T (number of test cases).
2. While T is greater than 0:
   * Read integer n1 (size of the first array).
   * Create array arr1 of size n1.
   * Fill arr1 with user input.
   * Read integer n2 (size of the second array).
   * Create array arr2 of size n2.
   * Fill arr2 with user input.
   * Call function find(arr1, n1, arr2, n2).
3. Define function find(arr1, n1, arr2, n2):
   * Initialize indices i = 0 and j = 0.
   * While both indices are within their respective array bounds:If
4. Print a newline after processing each test case.

**PROBLEM:**

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 |

**PROGRAM:**

#include <stdio.h>

void find(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, 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]);

}

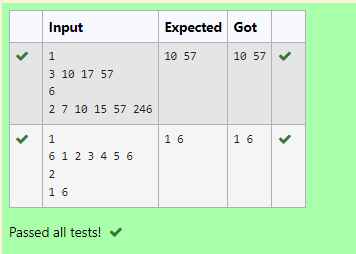
find(arr1, n1, arr2, n2);

}

return 0;

}

**OUTPUT:**

****

**AIM:**

Find the intersection of two sorted arrays.

**ALGORITHM:**

1. Read integer `T` (number of test cases).

2. While `T` is greater than 0:

- Read integer `N1` (size of the first array).

- Create array `arr1` of size `N1`.

- Fill `arr1` with user input.

- Read integer `N2` (size of the second array).

- Create array `arr2` of size `N2`.

- Fill `arr2` with user input.

- Call function `findIntersection(arr1, N1, arr2, N2)`.

3. Define function `findIntersection(arr1, n1, arr2, n2)`:

- Initialize indices `i = 0` and `j = 0`.

- While both indices are within their respective array bounds:

- If `arr1[i] == arr2[j]`, print the common element and increment both indices.

- If `arr1[i] < arr2[j]`, increment index `i`.

- Else, increment index `j`.

4. Print a newline after processing each test case.

**PROBLEM:**

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 |

**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++;

}

}

}

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);

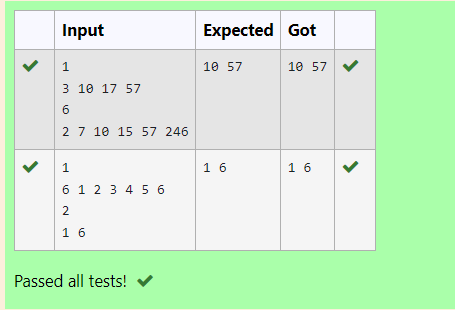
printf("\n");

}

return 0;

}

**OUTPUT:**

****

**AIM:**

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.

**ALGORITHM:**

1. Read integer `n` (size of the array).

2. Create array `arr` of size `n`.

3. Fill `arr` with user input.

4. Read integer `k` (the target difference).

5. Define function `diff(arr, n, k)`:

- Initialize indices `i = 0` and `j = 0`.

- While both indices are within the bounds of the array:

- Calculate `diff = arr[j] - arr[i]`.

- If `diff == k` and `i != j`, return `1` (indicating a valid pair is found).

- If `diff < k`, increment index `j`.

- Else, increment index `i`.

6. If no valid pair is found, return `0`.

7. In `main()`:

- Call `diff(arr, n, k)` and store the result.

- Print the result.

**PROBLEM:**

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 |

**PROGRAM:**

#include <stdio.h>

int diff(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++;

}

}

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 = diff(arr, n, k);

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

return 0;

}

**OUTPUT:**

****

**AIM:**

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.

**ALGORITHM:**

1. Read integer `n` (size of the array).

2. Create array `A` of size `n`.

3. Fill `A` with user input.

4. Read integer `k` (the target difference).

5. Define function `find(n, A, k)`:

- Initialize indices `i = 0` and `j = 0`.

- While index `j` is less than `n`:

- Calculate `d = A[j] - A[i]`.

- If `d == k` and `i != j`, return `1` (indicating a valid pair is found).

- If `d < k`, increment index `j`.

- Else:

- Increment index `i`.

- If `i` equals `j`, increment index `j`.

6. If no valid pair is found, return `0`.

7. In `main()`:

- Call `find(n, A, k)` and store the result.

- Print the result.

**PROBLEM:**

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 |

**PROGRAM:**

#include <stdio.h>

int find(int n, int A[], int k)

{

int i = 0, j = 0;

while (j < n)

{

int d = A[j] - A[i];

if (d == k && i != j)

{

return 1;

}

else if (d < k)

{

j++;

}

else

{

i++;

if (i == j)

{

j++;

}

}

}

return 0;

}

int main()

{

int n;

scanf("%d", &n);

int A[n];

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

{

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

}

int k;

scanf("%d", &k);

int result = find(n, A, k);

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

return 0;

}

**OUTPUT:**

****