

## Competitive programming:

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

**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 1 2 3 4	

```

1 #include <stdio.h>
2
3 int main() {
4     int n;
5     scanf("%d", &n);
6
7     int arr[n];
8
9     for (int i = 0; i < n; i++) {
10        scanf("%d", &arr[i]);
11    }
12
13    int duplicate = -1;
14
15    for (int i = 0; i < n; i++) {
16        for (int j = i + 1; j < n; j++) {
17            if (arr[i] == arr[j]) {
18                duplicate = arr[i];
19            }
20        }
21        if (duplicate != -1)
22            break;
23    }
24
25    printf("%d", duplicate);
26
27    return 0;
28 }
29

```

	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! ✓

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

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 1 2 3 4	

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2
3 int main() {
4     int n;
5     scanf("%d", &n);
6
7     int arr[n];
8     for (int i = 0; i < n; i++) {
9         scanf("%d", &arr[i]);
10    }
11
12    int duplicate = -1;
13    for (int i = 0; i < n; i++) {
14        for (int j = i + 1; j < n; j++) {
15            if (arr[i] == arr[j]) {
16                duplicate = arr[i];
17                break;
18            }
19        }
20        if (duplicate != -1)
21            break;
22    }
23
24    if (duplicate != -1)
25        printf("%d", duplicate);
26    else
27        printf("No duplicate found");
28
29    return 0;
30 }
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	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! ✓

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

**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:
  - Line 1 contains N1, followed by N1 integers of the first array
  - 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
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	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! ✓

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

**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  $N_1$ , followed by  $N_1$  integers of the first array
  2. Line 2 contains  $N_2$ , followed by  $N_2$  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	10 57
3 10 17 57	
6	
2 7 10 15 57 246	

```
1 #include <stdio.h>
2
3 int main() {
4     int T;
5     scanf("%d", &T);
6     while (T--) {
7         int n1, n2;
8         scanf("%d", &n1);
9         int arr1[n1];
10        for (int i = 0; i < n1; i++)
11            scanf("%d", &arr1[i]);
12        scanf("%d", &n2);
13        int arr2[n2];
14        for (int i = 0; i < n2; i++)
15            scanf("%d", &arr2[i]);
16        int i = 0, j = 0;
17        int found = 0;
18        while (i < n1 && j < n2) {
19            if (arr1[i] == arr2[j]) {
20                printf("%d ", arr1[i]);
21                found = 1;
22                i++;
23                j++;
24            } else if (arr1[i] < arr2[j]) {
25                i++;
26            } else {
27                j++;
28            }
29        }
30
31        if (!found)
32            printf("No common elements");
33
34        printf("\n");
35    }
36    return 0;
37}
38
```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	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! ✓

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

**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  $A[j] - A[i] = k$ ,  $i \neq j$ .

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
1 3 5	
4	

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

```
1 #include <stdio.h>
2 int main() {
3     int n, k;
4     scanf("%d", &n);
5     int arr[n];
6     for (int i = 0; i < n; i++) {
7         scanf("%d", &arr[i]);
8     }
9     scanf("%d", &k);
10    int i = 0, j = 1;
11    int found = 0;
12    while (i < n && j < n) {
13        if (i != j && arr[j] - arr[i] == k) {
14            found = 1;
15            break;
16        }
17        else if (arr[j] - arr[i] < k) {
18            j++;
19        }
20        else {
21            i++;
22        }
23    }
24    printf("%d", found);
25    return 0;
26 }
27 }
```

	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! ✓

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

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  $A[j] - A[i] = k$ ,  $i \neq j$ .

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
1 3 5	
4	

```

1 #include <stdio.h>
2 v int main() {
3     int n, k;
4     scanf("%d", &n);
5     int arr[n];
6 v     for (int i = 0; i < n; i++) {
7         scanf("%d", &arr[i]);
8     }
9     scanf("%d", &k);
10    int i = 0, j = 1;
11    int found = 0;
12 v    while (i < n && j < n) {
13        if (i != j && arr[j] - arr[i] == k) {
14            found = 1;
15            break;
16        } else if (arr[j] - arr[i] < k) {
17            j++;
18        } else {
19            i++;
20        }
21    }
22    printf("%d", found);
23    return 0;
24 }
25

```

	<b>Input</b>	<b>Expected</b>	<b>Got</b>	
✓	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! ✓