COMPETITIVE PROGRAMMING

Finding Duplicates-O(n^2) 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 2 3 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int find_duplicate(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                return 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]);
    }
    printf("%d\n", find_duplicate(arr, n));
    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	✓

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 2 3 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int findDuplicate(int nums[], int n) {
    int slow = nums[0];
    int fast = nums[0];
    do {
        slow = nums[slow];
        fast = nums[nums[fast]];
    } while (slow != fast);
    slow = nums[0];
    while (slow != fast) {
        slow = nums[slow];
        fast = nums[fast];
    }
    return slow;
}
int main() {
    int n;
    scanf("%d", &n);
    int nums[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }
    printf("%d\n", findDuplicate(nums, n));
    return 0;
}
```

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

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

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

```c
#include <stdio.h>
void findIntersection(int arr1[], int n1, int arr2[], int n2) {
    for (int i = 0; i < n1; i++) {
        for (int j = 0; j < n2; j++) {
            if (arr1[i] == arr2[j]) {
                printf("%d ", arr1[i]);
                break;
            }
        }
    }
    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;
}
```

	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! ✓

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

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++; // Move i forward
        } else if (arr1[i] > arr2[j]) {
            j++; // Move j forward
        } else { // arr1[i] == arr2[j]
            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]);
        }
        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! ✓

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

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 pairWithDifference(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);
    printf("%d\n", pairWithDifference(arr, n, k));
    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 8	0	0	✓
✓	10 0 2 3 7 13 14 15 20 24 25	1	1	✓

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

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int pairWithDifference(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++;
        }
    }
    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);
    printf("%d\n", pairWithDifference(arr, n, k));
    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.