COMPETITIVE PROGRAMMING

PROBLEM-1

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

ALGORITHM:

1. Read the integer `n`, which specifies the number of elements in the array.

2. Initialize an array of size `n` to store the input integers.

3. Loop through `n` iterations to read `n` integers and store them in the array.

4. Use a nested loop: the outer loop iterates through each element, and the inner loop compares the current element with subsequent elements.

5. If a duplicate element is found (i.e., the element in the outer loop equals the element in the inner loop), print that element.

6. The algorithm will print each duplicate element once, if any.

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

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 j=0;j<n;j++){
        for(int k=j+1;k<n;k++){
            if(arr[j]==arr[k]){
                printf("%d",arr[j]);
            }

        }
    }
}
```

OUTPUT:

	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	✔

PROBLEM-2

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

ALGORITHM:

1. Read the integer `n`, representing the size of the array, and initialize an integer variable `ind` for indexing and `m` for input values.

2. Create an array `arr` of size `n` to store values, initializing it with zeros or undefined values.

3. Iterate over `n` input values: for each value `m`, compute `ind = m % n` to determine the index position in the array.

4. If the value `m` already exists at the calculated index `arr[ind]`, print the value `m` as it is a duplicate and exit the loop.

5. If the value `m` is not found at the index, store `m` in the array at position `arr[ind]`.

6. The program detects and prints the first duplicate number encountered based on the modulo operation.

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,ind,m;
    scanf("%d",&n);
    int arr[n];
    for(int j=0;j<n;j++){
        scanf("%d",&m);
        ind=m%n;
        if (arr[ind]==m)
        {
            printf("%d",m);
            break;
        }
        else
        arr[ind]=m;
    }
}
```

OUTPUT:

	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	✔

PROBLEM-3

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

ALGORITHM:

1. Read the number of test cases `T`.

2. For each test case, read two arrays `arr1[]` and `arr2[]`.

3. Use two pointers to compare the arrays:

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

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

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

4. Repeat until all common elements are printed.

5. Print a newline after each test case.

PROBLEM:

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

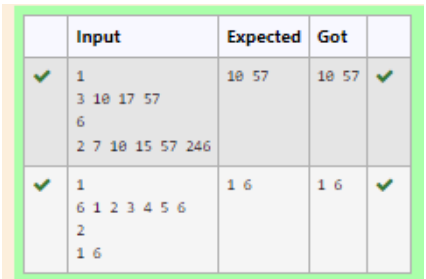
PROGRAM:

```
#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 :



	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	✔

PROBLEM-4

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

ALGORITHM:

1. Initialize two pointers `i` and `j` to 0.

2. Loop through both arrays until one is fully traversed.

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

4. If `arr1[i] > arr2[j]`, increment `j`.

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

6. Print a newline after all intersections are printed.

PROBLEM:

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

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]) {
            i++;
        } else if (arr1[i] > arr2[j]) {
            j++;
        } else {
            printf("%d ", arr1[i]);
            i++;
            j++;
        }
    }
    printf("\n");
}
```

OUTPUT:

	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	✔

PROBLEM-5

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

ALGORITHM:

1. Initialize two pointers `i` and `j` to 0.

2. Iterate through the array using the two pointers:

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

   - If `diff == k` and `i != j`, return 1 (pair found with the desired difference).

   - If `diff < k`, increment `j` to increase the difference.

   - If `diff > k`, increment `i` to decrease the difference.

3. If no pair is found, return 0.

4. In `main()`, read the array and the difference `k`, then call `findPairWithDifference` to find and 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.

PROGRAM:

```
#include <stdio.h>

int findPairWithDifference(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 = findPairWithDifference(arr, n, k);
    printf("%d\n", result);
    return 0;
}
```

OUTPUT:

	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	✔

PROBLEM-6

AIM: 6- Pair with Difference -O(n) Time Complexity,O(1) Space Complexity Completion requirements

ALGORITHM:

1. Initialize pointers `i` and `j` to 0.

2. Calculate the difference `arr[j] - arr[i]`.

3. If the difference equals `k`, return `1`.

4. If the difference is less than `k`, increment `j`.

5. If the difference is greater than `k`, increment `i`.

6. Return `0` if no valid pair is found.

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.

PROGRAM:

```
#include <stdio.h>

int findPairWithDifference(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 = findPairWithDifference(arr, n, k);
    printf("%d\n", result);
    return 0;
}
```

OUTPUT:

	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	✔