NAME : HARISH R P

REG.NO : 230701106

DEPT : B E COMPUTER SCIENCE AND ENGINEERING - B

Competitive Programming

a. Finding Duplicates-O(n^2) Time Complexity (1) Space Complexity

Aim: 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

Algorithm:

function main()

{

initialize n // Number of elements in the array read n from user

initialize arr[n] // Array to hold input values

// Read values into the array for i from 0 to n - 1

{

read arr[i] from user

}

flag = 0 // Initialize a flag to indicate if a duplicate is found

// Search for the first duplicate elementfor i from 0 to n - 1

{

el1 = arr[i] // Current element

for j from 0 to n - 1

{

// Check for duplicates and ensure indices aredifferent if el1 == arr[j] and i != j

{

print el1 // Print the duplicate element

flag = 1 // Set flag to indicate a duplicate was found break

```
// Exit inner loop
}
}


if flag
break // Exit outer loop if a duplicate was found
}
}
```

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]);
}


int flag=0;
for(int  i=0;i<n;i++){int el1=arr[i];

for(int j=0;j<n;j++){if (el1==arr[j] &&
i!=j){
```

```
printf("%d",el1);flag=1;

break;


}

}


if(flag

)

break

;




}

}
```

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	✔

a. Finding Duplicates-O(n) Time Complexity (1) Space Complexity

Aim: 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

Algorithm:

function main()

{

initialize n // Number of elements in the array read n from user

initialize a[n] // Array to hold input values

// Read values into the array for i from 0 to n - 1

{

read a[i] from user

}

initialize b[n] // Array to keep track of seen elements for i from 0 to n - 1

{

b[i] = 0 // Initialize the tracking array

}

// Search for the first duplicate elementfor i from 0 to n - 1

{

// If the element is already present, i.e., b[a[i]] = 1 if b[a[i]]

{

print a[i] // Print the duplicate elementbreak // Exit the loop

}

else

{

b[a[i]] = 1 // Mark the element as seen

}

}

}

Program:

#include

```
<stdio.h> intmain(){

int n; scanf("%d",&n);int a[n];

for(int i=0;i

<n;i++){

scanf("%d",&a[i]

);



}



int b[n];



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

}

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

//if el already present i.e, b[i]=1if(b[a[i]]){

printf("%d",a[i]);break;

}

else b[a[i]]=1;

}







}
```

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	✔

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

Aim:

Find the intersection of two sortedarrays. 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 thesecond array Output Format

The intersection of the arrays in a singleline 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

Algorithm:

function main()

{

initialize n // Number of testcases read n from user

for i from 0 to n - 1

{

initialize n1 // Size of the firstarray read n1 from user

initialize arr1[n1] // First array

// Read values into the first arrayfor j from 0 to n1 - 1

{

read arr1[j] from user

}

```
initialize n2 // Size of the second array read n2 from user

initialize arr2[n2] // Second array

// Read values into the second array for j from 0 to n2 - 1
{

read arr2[j] from user
}

// Check for common elements in both arrays for j from 0 to n1 - 1
{

for k from 0 to n2 - 1
{

if arr1[j] == arr2[k]
{

print arr1[j] // Print the common element
}
}
}
```

```
}
}
```

Program: #include<stdio.h> int main(){

```
int n; scanf("%d",&n); for(int i=0;i<n;i++){
int n1; scanf("%d",&n
```

1. ; int arr1[n1]; for(int j=0;j<n1;j++){

```
scanf("%d ",&arr1[j]);

}

int n2; scanf("%d",&n2);

int arr2[n2];
for(int j=0;j<n2;j++){ scanf("%d ",&arr2[j]);
}
for(int j=0;j<n1;j++){ for(intk=0;k<n2;k++){
if(arr1[j]==arr2[k]){

printf("%d ",arr1[j]);
}
}
}
}
```

}

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	✔

a. **Print Intersection of 2 sorted arrays-O(m+n)Time Complexity,O(1) SpaceComplexity**

Aim:

Find the intersection of two sortedarrays. 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 thesecond 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

Algorithm:

function main()

{

initialize T // Number of test cases read T from user

while T > 0

{

// Decrement the test case counter T--

initialize n1, n2 // Sizes of the two arrays read n1 from user

initialize arr1[n1] // First array

// Read values into the first array for i from 0 to n1 - 1

{

read arr1[i] from user

}

read n2 from user

initialize arr2[n2] // Second array

// Read values into the second array for i from 0 to n2 - 1

{

read arr2[i] from user

}

initialize i = 0, j = 0 // Indices for both arrays

// Iterate through both arrays to find common elements while i < n1 and j < n2

{

```
if arr1[i] < arr2[j]
{

i++ // Move to the next element in arr1
}

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

j++ // Move to the next element in arr2
}

else
{

print arr1[i] // Print the common element i++
// Move to the next element in arr1 j++ //Move to the next element in arr2
}
}



print new line // Move to the next line for output
}
}
```

Program:

```
#include <stdio.h>

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]);
}

int i = 0, j = 0;
while (i < n1 && j < n2) { if(arr1[i] < arr2[j]) {
i++;

}
```

```
else if (arr2[j] < arr1[i]) { 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	✔

a. Pair with Difference-O(n^2)TimeComplexity,O(1) Space Complexity

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.

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.

Algorithm:

function main()

{

initialize n // Number of elements in the array read n from user

initialize arr[n] // Array to hold input values

// Read values into the array for i from 0 to n - 1

{

```
read arr[i] from user

}

initialize t // Target difference read t from user

initialize flag = 0 // Flag to indicate if a pair is found

// Check for pairs with the specified difference for i from 0 to n - 1

{

for j from 0 to n - 1

{

if i != j and abs(arr[i] - arr[j]) == t

{

flag = 1 // Pair found break

}

}

if flag

{
```

```
break

}

}



// Output the result based on theflag if flag

{


print 1 // Pair found



}

else

{


print 0 // No pair found

}



return 0

}
```

Program:

```
#include

<stdio.h> #include

<stdlib.h>
```

```
int main()

{ int n; scanf("%d", &n);


int arr[n];




for (int i = 0; i < n; i++) { scanf("%d",&arr[i]);

}



int t; scanf("%d",&t);


int flag = 0;



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

{


if (i!=j && abs(arr[i] - arr[j]) == t) {flag = 1;

break;


}

}
```

```
if (flag)

{

break

;



}

}
```

```
if (flag) {

printf("%d\n", 1);

} else {


printf("%d\n", 0);

}
```

```
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	✔

a. Pair with Difference -O(n) TimeComplexity,O(1) Space Complexity

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.

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.

Algorithm:

function main()

{

initialize n // Number of elements in the arrayread n from user

initialize arr[n] // Array to hold input values

// Read values into the array for i from 0 to n - 1

{

read arr[i] from user

}

initialize t // Target differenceread t from user

initialize flag = 0 // Flag to indicate if a pair is found

initialize i = 0 // First index initialize j = 1 // Second index

// Loop to find pairs with the specified differencewhile i < n and j < n

{

diff = abs(arr[i] - arr[j]) // Calculate the difference

```
if i != j and diff == t

{


flag = 1 // Pairfound break

}


else if diff < t

{


j++ // Increment second index

}


else

{


i++ // Increment first index

}

}




// Output the result based on theflag if flag

{


print 1 // Pair found

}
```

else

{

print 0 // No pair found

}

return 0

}

Program:

#include

<stdio.h> #include

<stdlib.h>

int main()

{ int n; scanf("%d", &n);

int arr[n];

for (int i = 0; i < n; i++) { scanf("%d",&arr[i]);

```
}



int t; scanf("%d",&t);


int flag = 0;



int i=0;

int j=1;

while(i<n && j<n){

int diff = abs(arr[i] - arr[j]);if(i!=j && diff==t){

flag=1;

break

;


}

else if(diff<t){j++;

}

else{

i++;


}



}


}
```

```
if (flag) {
printf("%d\n", 1);
} else {

printf("%d\n", 0);

}

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	✔