Ques. Increasing Order

smallest = 3
largest = 5

Quiz 1.

3, 4, 4, 4, 4, 4, 4, 5, 5, 5

previous <= current

Yes, it is in increasing order

Quiz 2 →

-1 -2 -3 -4 -5

No.

smallest → -5
largest → -1

-5 -4 -3 -2 -1

Quiz 3-

3, 4, 4, 4, 4, 4, 4, 5, 5, 5

not in Strictly inc. order

Increasing order

3, 3, 5, 9, 15, 21, 21

previous <= current

Strictly increasing order

3, 5, 17, 24, 51

prev < current

Ques. Given an array, return true if array is in strictly increasing order otherwise return false.

ans

A = { 3, 5, 19, 18, 21 } ⟶ false

A = { 3, 5, 9, 9, 15 } ⟶ false

A = { -1, 15, 24, 25, 90 } ⟶ true
0 1 2 3 4

prev index = 0
2
(i-1)

current index = 1
3
i

prev element = A[i-1]

current element = A[i]

Wrong

```
for(int i=1; i < A.length; i++){
    if( A[i] > A[i-1] ){
        return true;
    }
}
return false;
```

correct

```
for( int i=1; i<A.length; i++){
    if(A[i] > A[i-1]){
        // do nothing
    }
    else {
        return false;
    }
}
return true;
```

[not in strictly inc order]

A = { 1, 2, 4, 4, 5 }
0 1 2 3 4

⟶ ans = false

i	i < 5	A[i] > A[i-1]	i++
1	true	2 > 1 (true)	⟶ return true

true → for all indexes from 1 to n-1 → A[i] > A[i-1]

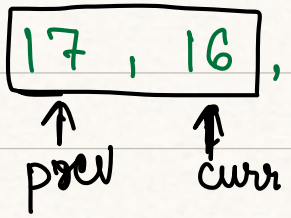
⇓

If we find even one index which is not following this condition, answer will be false

curr > prev → true

curr < prev or curr == prev → false

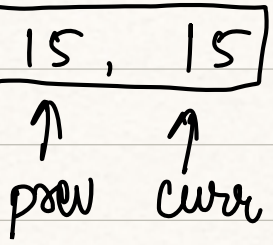
{3, 4, 9, 17, 16, 25}



curr < prev

Hence, ans is false

{3, 5, 9, 15, 15, 17}



curr == prev

Hence, ans is false

If curr <= prev then array is not in strictly increasing order

```
for ( int i = 1 ; i < A.length ; i++) {

    if ( A[i] <= A[i-1] ) {
        return false;
    }

}

return true;
```

A = { 3, 9, 17, 100 } → true

(indices: 3 → 0, 9 → 1, 17 → 2, 100 → 3)

i	i < 4	A[i] <= A[i-1]	i++
1	true	9 <= 3 false	2
2	true	17 <= 9 false	3
3	true	100 <= 17 false	4
4	false →	break	

Homework

check if given array is in increasing order

Ques. Given an array containing each element exactly twice except one element.

Return the element which occurs only once.

A = { 3, 5, 9, 5, 3, 6, 9 } ⇒ ans 6

frequency

3	5	9	5	3	6	9
↓	↓	↓	↓	↓	↓	↓
2	2	2	2	2	1	2

A = { 6, 9, 1, 2, 9, 2, 6 } ⇒ 1

① count the frequency of element

② find the element with frequency = 1.

```
int countFreq(int[] A, int k) {
    int count = 0;
    for(int i = 0; i < A.length; i++) {
        if (A[i] == k) {
            count++;
        }
    }
    return count;
}
```

```
int findUniqueElement(int[] A) {
    int ans = 0;
    for (int i=0; i<A.length; i++) {
        int freq = countFreq(A, A[i]);
        if (freq == 1) {
            ans = A[i];
            break;
        }
    }

    return ans;
}
```

Nested Loops

A = { 6, 9, 1, 2, 9, 2, 6 }

0 1 2 3 4 5 6

```
for ( int i = 0; i < A.length; i++ ) {
    // count freq. of ith index element (A[i])

    int count = 0;
    for ( int j = 0; j < A.length; j++ ) {
        if ( A[j] == A[i] ) {
            count++;
        }
    }
    if ( count == 1 ) {
        ans = A[i];
        break;
    }
}
return ans;
```

0 1 2 3 4 5 6

{ 6, 9, 1, 2, 9, 2, 6 }

$i = 0$ $A[0] = 6$

Count = 0

j	$j < 7$	$A[j] == 6$	count
0	✓	✓	1
1	✓	X	1
2	✓	X	1
3	✓	X	1
4	✓	X	1
5	✓	X	1
6	✓	✓	2

$i = 4$ $A[4] = 9$

Count = 0

$j = 4$

Ques. Given an array and a value K. Return true if there is any pair such that A[i] + A[j] == K (i != j) otherwise return false.

A = {3, 5, 9, 1, 2, 4}
0 1 2 3 4 5

K = 10
true
i = 2 j = 3
(9, 1)

K = 7
true
(3, 4) i = 0 j = 5
(5, 2) i = 1, j = 4

Approch 1 →

① check all the pair (i, j)

② If (i != j) then check whether sum of that pair is equal to K or not

K = 17
false

K = 18
false

$A = \{3, 5, 1, 4\}$

indices: 0, 1, 2, 3

Total pairs = 16

i	j
0	0, 1, 2, 3
1	0, 1, 2, 3
2	0, 1, 2, 3
3	0, 1, 2, 3

(0,0)	(0,1)	(0,2)	(0,3)
(1,0)	(1,1)	(1,2)	(1,3)
(2,0)	(2,1)	(2,2)	(2,3)
(3,0)	(3,1)	(3,2)	(3,3)

(1, 0) ⟶ $A[1] + A[0] = 5 + 3 = 8$

(0, 1) ⟶ $A[0] + A[1] = 3 + 5 = 8$

$A[i] + A[j]$ $A[j] + A[i]$

$A[i] + A[j] == K$ && $(i \,!= j)$

```
// check all the pairs ⟶ (16 pairs)
for ( int i = 0 ;  i < A.length ;  i++) {
    for ( int j = 0 ;  j < A.length ;  j++) {
        if ((i != j)  &&  (A[i] + A[j] == K)) {
            return true;
        }
    }
}
return false;
```

Better approach →

(0,0) (0,1) (0,2) (0,3)	
(1,0) (1,1) (1,2) (1,3)	
(2,0) (2,1) (2,2) (2,3)	
(3,0) (3,1) (3,2) (3,3)	

(i , j)

i	j
0	1, 2, 3
1	2, 3
2 [second last index]	3

$i = 0 \rightarrow (n-2)$ $\quad j = (i+1) \rightarrow (n-1)$

// checking only 6 pairs

More optimised

```
for (int i = 0; i <= A.length-2; i++) {
    for (int j = (i+1); j <= A.length-1; j++) {
        if (A[i] + A[j] == k) {
            return true;
        }
    }
}
return false;
```

$A = \{3, 5, 1, 4\}$
0 1 2 3

$K = 9$
(true)

i		j	A[i]+A[j]==9
0	(0<=2)	1 ✓	3+5 == 9 X
		2 ✓	3+1 == 9 X
		3 ✓	3+4 == 9 X
		4 X break	
1	(1<=2)	2 ✓	5+1 == 9 X
		3 ✓	5+4 == 9 ✓

→ return true

First & Last Index

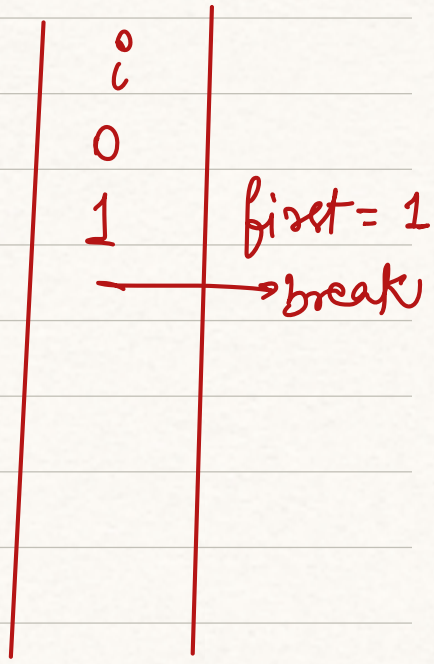
A = { 1, 9, 9, 9, 10, 21 } B = 9

0 1 2 3 4 5

int[] ans = | 1 | 3 |

0 1

```
int first = 0;
for(int i = 0; i < A.length; i++){
    if (A[i] == B){
        first = i;
        break;
    }
}
```



A = { 1, 9, 9, 9, 10, 21 } B = 9

0 1 2 3 4 5

```
int last = 0

for (int i = 0 ; i < A.length ; i++) {
    if (A[i] == B) {
        last = i;
    }
}

int [] ans = new int [2];

ans [0] = first ;
ans [1] = last ;

return ans;
```

i	
0	
1	last = 1
2	last = 2
3	last = 3
4	
5	
6	→ break

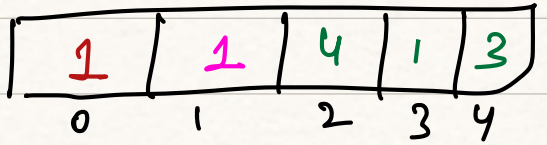
MCQ-F

```
int [] arr = {0, 2, 4, 1, 3};
```

0 1 2 3 4

```
for (int i = 0; i < arr.length; i++){
    arr[i] = arr[ arr[i] + 3 % arr.length];
}
```

arr



i = 0

arr[0] = arr[arr[0] + 3 % 5];

= arr[0 + 3 % 5]

= arr[3 % 5]

arr[0] = arr[3];

i = 1 arr[i] = arr[arr[i] + 3 %5];

= arr[2 + 3 %5];

= arr[5 %5];

arr[1] = arr[0];

arr[1] = 1