Q.1 Given an Array A[] of size N and a value K. Write a function that returns true is K is present in A[] otherwise returns false.

	0	1	2	3	4	5
A :	3	2	4	9	1	8

K = 9 ⟶ true

K = 17 ⟶ false

```
boolean search (int [] arr, int k) {
    boolean isfound = false;
    int n = arr.length;
    for (int i=0; i<n; i++) {
        if (arr[i] == k) {
            isfound = true;
            break;
        }
    }
    return isfound;
}
```

K = 17

isfound = false

i

	0	1	2	3	4	5
	3	2	4	9	1	8

```
boolean search (int [] arr, int k) {
    int n = arr.length;
    for (int i=0; i<n; i++) {
        if (arr[i] == k) {
            return true;
        }
    }
    return false;
}
```

```
main () {
    == ] -> input
    boolean ans = search(arr, k);
}
```

Q.2 Given an Array A[] and element K. Write a function that returns count of K in A[].

	0	1	2	3	4	5	6	7	8	9
A :	3	2	9	6	3	4	9	3	2	3

K = 3 ⟶ 4

K = 9 ⟶ 2

K = 11 ⟶ 0

```
int countOfEle (int[] arr, int K) {
    int count = 0;
    for(int i=0; i < arr.length; i++) {
        if (arr[i] == K) {
            count++;
        }
    }
    return count;
}
```

K = 3

count = ~~0 1 2 3~~ 4

i

	0	1	2	3	4	5	6	7	8	9
	3	2	9	6	3	4	9	3	2	3

Q.3 Given an Array A[]. Write a function which returns true if array is arranged in increasing order (Strictly) otherwise returns false.

A: 3 4 6 10 12 ⟶ true

A: 3 9 12 12 15 ⟶ false

A: 3 9 10 8 14 ⟶ false

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

conclusion: for array to be in inc. order all adjacent pairs must follow a[i] > a[i-1]

if there is an adjacent pair where a[i] <= a[i-1] return false.

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

A:

0	1	2	3	4
3	4	6	5	12

i	i-1	
1	0	$4 \leq 3$
2	1	$6 \leq 4$
3	2	$5 \leq 6$ false
4	3	

```
boolean increasing (int [] arr) {
    for(int i = 0 i < arr.length-1; i++) {
        if (arr[i] >= arr[i+1]) {
            return false;
        }
    }
    return true;
}
```

0	1	2	3
3	4	5	7

i	i+1	
0	1	$3 \geq 4$
1	2	$4 \geq 5$
2	3	$5 \geq 7$
3	4	$7 \geq$ ⚠

10 students

```
int [] arr = new int [10];
// take input in array
// 5 new admission
int [] new_array = new int [15];
// firstly copy 10 values from arr to new_arr and
   take 5 new inputs from user.
```

Array lists : Arrays only but with size modification.

```
ArrayList < Integer > al = new ArrayList <> ( );
```

datatype — name

```
void main() {
    Scanner scn = new Scanner();
    ArrayList<Integer> al = new ArrayList<>();
    int N = scn.nextInt();
    for(int i=1; i<=N; i++) {
        int val = scn.nextInt();
        al.add(val);
    }
    // print ArrayList
    for (int i = 0; i<N; i++) {
        SOP(al.get(i));
    }
    // add 3 more values to arraylist
    for(int i=0; i<3; i++) {
        int val = scn.nextInt();
        al.add(val);
    }
    SOP(al);
    al.remove(3);   // index
    SOP(al);
}
```

Q.4 Given an ArrayList in which all the elements are present two times except one element. Find that single element.

Al : 3 8 3 4 9 4 9

Al : 3 3 8 6 8

```
void main() {
    Scanner scn = new Scanner();
    int N = scn.nextInt();
    ArrayList<Integer> al = new ArrayList<>();
    // take input
    for (int i = 1; i <= N; i++) {
        int val = scn.nextInt();
        al.add(val);
    }
    SOP (singleEle(al));
}
```

```
int singleElement(ArrayList<Integer> al) {
    for(int i=0; i<al.size(); i++) {
        int ele = al.get(i);
        int count = 0;
        for(int j=0; j<al.size(); j++) {
            if(al.get(j) == ele) {
                count++;
            }
        }
        if(count==1) {
            return ele;
        }
    }
    return 0; // never executes if input is correct.
}
```

3	3	8	6	8
0	1	2	3	4

i	j		
0	[0,4)	ele = 3	c = 0 x 2
1	[0,4]	ele = 3	c = 0 x 2
2	[0,4]	ele = 8	c = 0 x 2
3	[0,4]	ele = 6	c = 1
4			