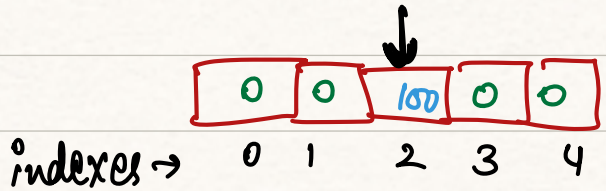
```
int[] arr = new int[5];
```

↳ size




```
arr[2] = 100;
```

n

indexes → 0 - (n-1)

```
for(int i=0 ; i< arr.length; i++){

    arr[i] = scn.nextInt();

}
```

i = 0	arr[0]
1	arr[1]
2	arr[2]
⋮	⋮

Quiz 1.

A = [3, 5, 1, 6, 2, 4, 7]

(indices: 0 1 2 3 4 5 6)

A[2] + A[3] = 7

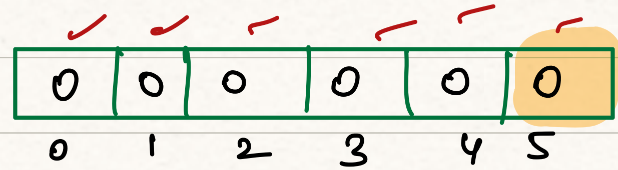
A[2] = 1

A[3] = 6

Quiz 2.

int A[] = new int[6] → size

S.O.P (A[5] + A[6]);



A[5] = 0

A[6] => Error

Array Index Out of Bound

Ques. Given an array A of size n and a value 'k'

If k is present in array return true
else return false.

Search in array

A = { 2, 9, 6, 15, 4, 10 }
0 1 2 3 4 5

n = 6

	ans
K = 19	false
K = 6	true

```
boolean search ( int[] A , int k ){

    for (int i=0 ; i< A.length ; i++) {
        if (A[i] == K){
            return true;
        }
```

}

return false;

}

true

A = { 2, 9, 6, 15, 4, 10 }
0 1 2 3 4 5

K = 9

i	i < 6	A[i] == 9	i++
0	true	2 == 9 (false)	1
1	true	9 == 9 (true) →	return true

A = { 2, 9, 6, 15, 4, 10 }
0 1 2 3 4 5

K = 3

[false]

i	i < 6	A[i] == 3	i++
0	true	2 == 3 false	1
1	true	9 == 3 false	2
2	true	6 == 3 false	3
3	true	15 == 3 false	4
4	true	4 == 3 false	5

5 true 10 == 3 false 6

6 false → break

return false.

Search in Array →

① Check and compare all the elements of array with k

② If they are equal return true

③ If no element is equal return false.

Ques. Given and array and k. Find the frequency of k in this array

A = { 2, 3, 9, 4, 3, 6, 4, 2, 1, 2 }

	ans
K = 2	3
K = 4	2
K = 17	0

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

        if (A[i] == k) {
            count ++;
        }
    }
    return count;
}
```

ans
2

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

K = 2

i	i < 5	A[i] == 2	count	i++
0	true	2 == 2 true	1	1
1	true	1 == 2 false	1	2
2	true	4 == 2 false	1	3
3	true	2 == 2 true	2	4
4	true	5 == 2 false	2	5
5	false	→ break		

count = 2

return 2

this code will not work if all the elements are negative

```
int findmax ( int [] A ) {
    int max = 0;
    for ( int i = 0 ; i < A.length ; i++ ) {
        if (A[i] > max) {
            max = A[i];
        }
    }
    return max;
}
```

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

ans
9

max = 0

i = 0	2 > 0	✓	max = 2
i = 1	9 > 2	✓	max = 9
i = 2	5 > 9	X	max = 9
i = 3	4 > 9	x	max = 9

max = 0

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

ans = -1

i = 0	-4 > 0	x
1	-9 > 0	x
2	-5 > 0	x
3	-1 > 0	x

max = 0
max = 0
max = 0
max = 0

first element

int max = A[0];

Assuming that 0th index element is my maximum value

- correct → correct ans
- false → correct ans

Assumption is correct

A = { 100, 5, 9, 4 }
0 1 2 3

max = 100

i = 0	100 > 100	max = 100
1	5 > 100	max = 100
2	9 > 100	max = 100
3	4 > 100	max = 100

Assumption is false

A = { -1, 5, 15, -9 }
0 1 2 3

max = -1

i = 0	-1 > -1	max = -1
1	5 > -1	max = 5
2	15 > 5	max = 15
3	-9 > 15	max = 15

```
int findMaximum(int[] A){
    int max = A[0];
    for(int i = 1; i < A.length; i++){
        if (A[i] > max){
            max = A[i];
        }
    }
    return max;
}
```

Ques. Given an array and k. Return true if there is any adjacent pair with difference k otherwise return false.

index i → adjacent element are (i-1) and (i+1)

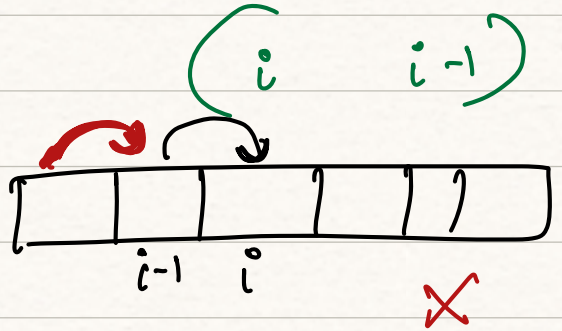
$A[i] - A[i+1] = k$, for any index

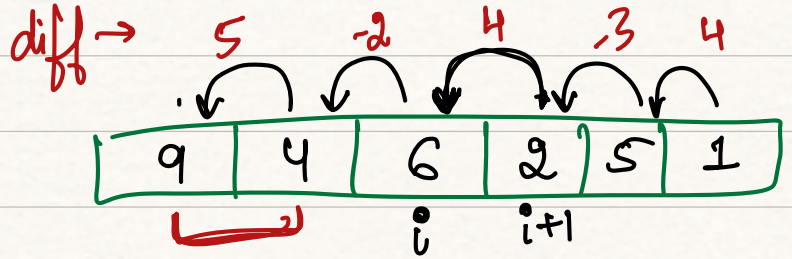
k = 4 true

A = { 9, 2, 8, 4, 6 }
0 1 2 3 4

i	A[i]	i+1	A[i+1]	A[i] - A[i+1]
0	9	1	2	9 - 2 = 7 ✓
1	2	2	8	2 - 8 = -6 ✓
2	8	3	4	8 - 4 = 4 ✓
3	4	4	6	4 - 6 = -2 ✓

K = 5 true
K = 6 false

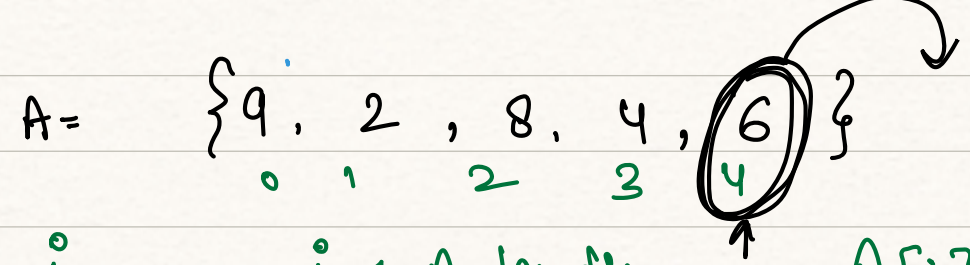




A[i] - A[i+1]
9 - 4

```
boolean adjDiff (int[] A , int K){
    for (int i=0 ; i < A.length -1 ; i++){
        if (A[i] - A[i+1] == K){
            return true;
        }
    }
    return false;
}
```

same [i+1 < A.length
       i < A.length -1



$k = 4$ true

i	i < A.length	A[i] - A[i+1] == 4	i++
0	0 < 5 ✓	9 - 2 == 4 X	1
1	1 < 5 ✓	2 - 8 == 4 X	2
2	2 < 5 ✓	8 - 4 == 4 ✓ ⟶ true	

$k = 1$ false

i	i < A.length	A[i] - A[i+1] == 1	i++
0	true	A[0] - A[1] = 9 - 2 = 7 (false)	1
1	true	A[1] - A[2] = 2 - 8 = -6 (false)	2
2	true	A[2] - A[3] = 8 - 4 = 4 (false)	3
3	true	A[3] - A[4] = 4 - 6 = -2 (false)	4
4	true	A[4] - A[5] = 6	

Error ⟶ Array Index Out of Bound

$i < A.length - 1$ ⟶ correct condition

Doubts

Make it

A = count of mangoes
B = count of slices

1 mango = 3 slices

 = 2 slice

2 mango = $2*3$ = 6 slices

A = 2
B = 3

A mango = $3*A$ slices

total slices = 6 + 3 = 9 slices

2 + 2 + 2 + 2 + 1

9 / 2 = 4

A = 7 ↓ B = 1 ↙

total slices = 21 + 1 = 22

[3 * A + B]

total shakes = $\frac{22}{2}$ = 11 .

A = Rs 50

B = Rs 5 ↓ one choc price

total choc = 10 choc

10

C = 15

C = 7

ans : 7

Multi-test

no. of test cases

for every test case { size of array, elements of array }

1
5
1 2 3 4 5

1 test case

size of array
elements of array

5
1 2 3 4 5

MCQ F

arr | 10 | 20 | 30 |

int [] arr = {10, 20, 30};

arr = new int [3]; ⟶ | 0 | 0 | 0 |

ans = arr[0] * arr[1];

ans ⟶ 0

```
void fun (int a) {
    S.O.Pln("World");
}

main ( ) {
    S.O.P ("Hello");
    fun ( );
}
```

Error

final int a = 10;

cannot change value of variable

// function that prints "Happy New Year"

```
void fun ( int n ) {
    S.O.Pln(n);
    // S.O.Pln("Happy New Year");
}

main () {
    → S.O.Pln("1");
    → fun (2);
    → S.O.Pln ("2");
}
```

Output

1
2
Happy New Year
2