arr[5] = { 1 2 -1 4 5 }

0 1 2 3 4

arr[0] + arr[4] = 6

Time complexity of accessing. ∀ O(1)

Print all elements of an Array.

```
int arr[];

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

    print(arr[i]);

}
```

TC: O(n)

Q1 Given N array elements, count number of elements having atleast 1 element greater than itself.

Eg arr[7] = { -1 2 4 6 5 6 2 }

ans = 5

Eg arr[6] = { 1 9 14 22 4 6 }

ans = 5

Observation

1) Max value would not be counted.
2) Find max value
3) Find count of max value.
4) ans ⇒ Size of Array - count of max value.

Pseudo Code

1) Find max value

Lowest Integer value

```
int max_val = arr[0] ; / INTEGER.MIN ;

for (int i=0 ; i<n ; i++) {
    if (arr[i] > max_val)
        max_val = arr[i]
}
```

Tc: O(n)
Sc: O(1)

2) Find count of max value elements

```
int c = 0

for (int i=0 ; i<n ; i++) {
    if (arr[i] == max_value)
        c++;
}
```

Tc: O(n)
Sc: O(1)

3) Return (n - c);

Tc: O(n)
Sc: O(1)

Q2 Given N array elements, print true if there exists a pair of index i & j, where arr[i] + arr[j] = k.

Note: 1) Array is not sorted
2) i != j
3) return true / false
4) k is given.

Eg 1 : arr[] = { 3 5 -1 6 4 }
0 1 2 3 4
k = 8

= true

Eg 2 : arr[] = { 2 4 3 -2 }
k = 8

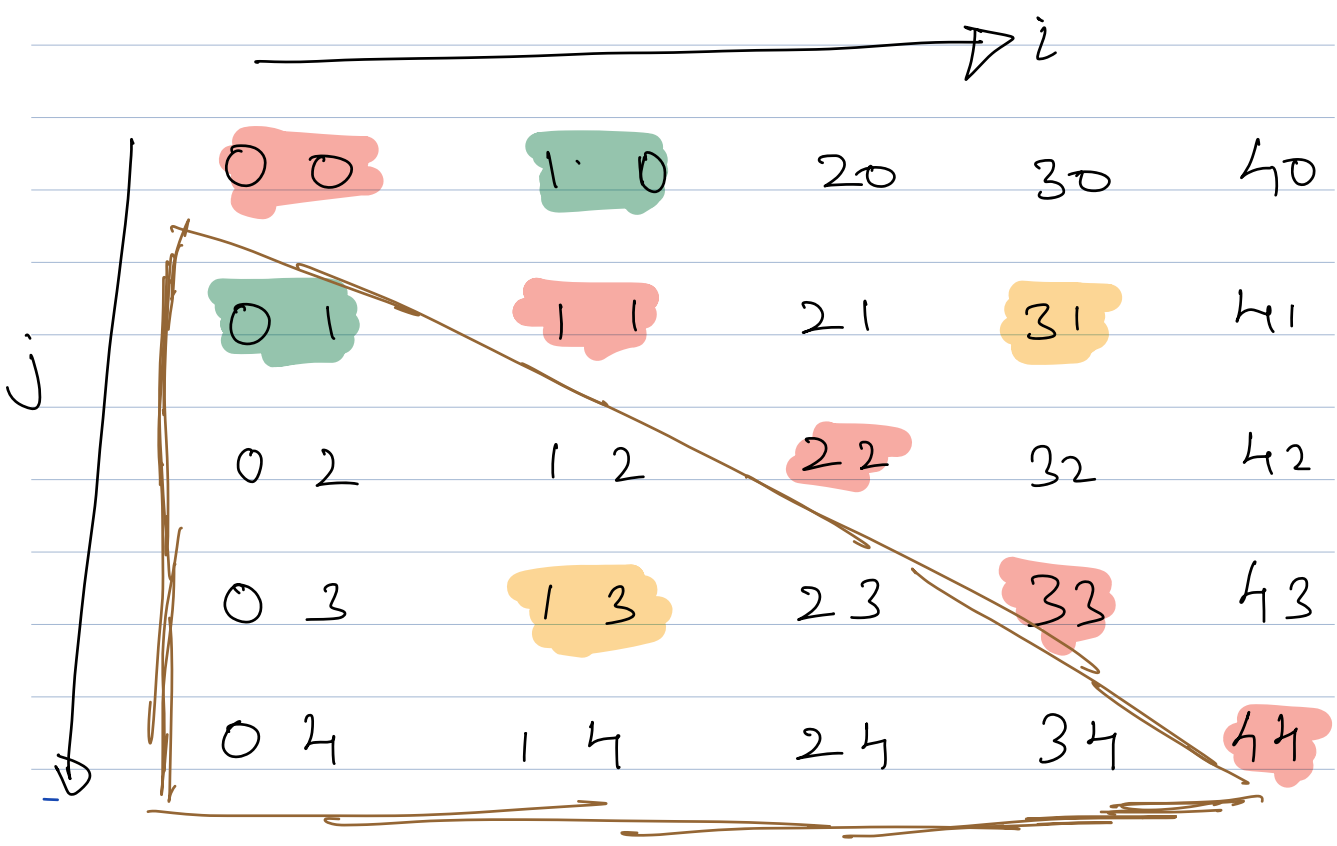
= false.

Pseudo Code

```
for (int i=0 ; i<n ; i++) {
    for (int j=0 ; j<n ; j++) {
        if ((i != j) && (arr[i] + arr[j] == k))
            return true
    }
}
return false;
```

Tc : $O(n^2)$

Sc : $O(1)$



```
for (int i=0 ; i<n ; i++) {
    for (int j=i+1 ; j<n ; j++) {
        if (arr[i] + arr[j] == k)
            return true
    }
}
```

TC: $O(n^2)$

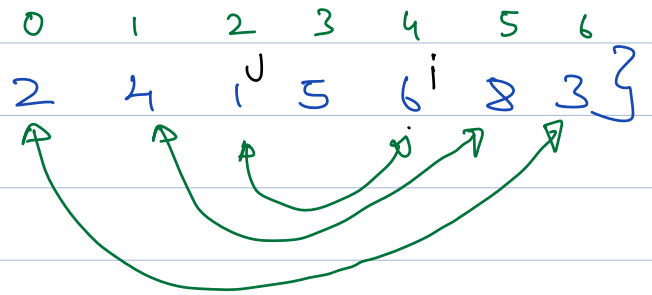
i	j	total iteration
0	[1, n-1]	n-1
1	[2, n-1]	n-2
:		n-3
:		
:		
n-2	[n-1, n-1]	1
n-1	X	0

$$0 + 1 + 2 + 3 + 4 + \dots + (n-1)$$

$$\frac{n(n+1)}{2} = \frac{(n-1)(n)}{2} \Rightarrow O(n^2)$$

Q3 Given an array. Reverse it.
SC should be O(1).

Ex1 arr[] :- { 2 4 1 5 6 8 3 }



arr[] = { 3 8 6 5 1 4 2 }

Terminating Condition.

		Odd Size	Even Size
1)	i == j	✓	✗
2)	i > j	✓	✓

Pseudo i=0, j=n-1

1)
```
while (j > i) {
    // swap
    i++;
    j--;
}
```

2)
```
while (j ≥ i) {
    // swap
    i++;
    j--;
}
```

Tc: O(n), Sc: O(1)

Q4 Reverse the array from start to end.

→ start and end indices are given.

→ start < end.

Eg :

	0	1	2	3	4	5	6
arr[] =	-2	4	6	7	8	1	5

start = 2	-2	4	1	8	7	6	5
end = 5							

```
void reverse (int start, int end, int arr[]) {

    int i = start;
    int j = end;

    while (j > i) {

        // swap
        i++;
        j--;
    }

}
```

TC : O(n)

SC : O(1)

Q5 Given n array elements, rotate the array from last to first k times.
SC should be O(1).

→ k < n

→ (0, 3) (0, n-k)

0 1 2 3 4 5 6 7-3

Eg 1 arr[] = 5 2 3 -1 6 1 2 (4)

k = 3

k = 1 2 5 2 3 -1 6 1

k = 2 1 2 5 2 3 -1 6

k = 3 6 1 2 5 2 3 -1

0 1 2 3 4 5 6

5 2 3 -1 6 1 2

↓ reverse first 4 elements. (0, n-1-k)

-1 3 2 5 6 1 2

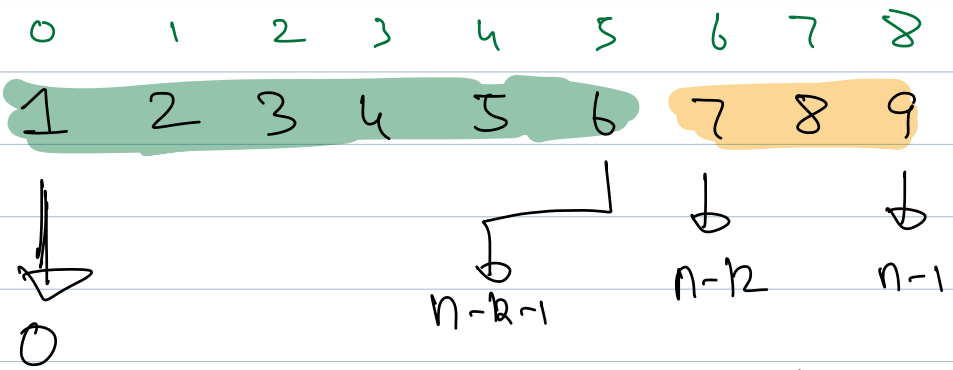
↓ reverse last 3 elements

-1 3 2 5 2 1 6

↓ reverse the entire array.

6 1 2 5 2 3 -1

k=3



$[s_1, n-1] \Rightarrow k$

b-a+1

$n-1-s_1+1 \Rightarrow k$

$n-k = s_1$

Pseudo code

```
reverse ( 0, n-k-1, arr);
reverse ( n-k, n-1, arr);
reverse ( 0, n-1, arr );
```

TC : O(n);

SC : O(1);

CASE 2

$k \geq n$

$n = 5$

$arr[] = \{a_1, a_2, a_3, a_4, a_5\}$

$k = 0, 5, 10, 15$	$a_1\ a_2\ a_3\ a_4\ a_5$
$k = 1, 6, 11, 16$	$a_5\ a_1\ a_2\ a_3\ a_4$
$k = 2, 7, 12, 17$	$a_4\ a_5\ a_1\ a_2\ a_3$
$k = 3, 8, 13, 18$	$a_3\ a_4\ a_5\ a_1\ a_2$
$k = 4, 9, 14, 19$	$a_2\ a_3\ a_4\ a_5\ a_1$
$k = 5$	$a_1\ a_2\ a_3\ a_4\ a_5$

$$k = k \% n$$

Dynamic Arrays ⇒ Where the size of array is not.

lang	C++	JAVA	Python	JS	C#
Dynamic Array	vector	ArrayList	list	Array	ArrayList

C ⇒ Change your language to C++.

⇒ list<int> l;

⇒ l.insert(10);

⇒ l.size()