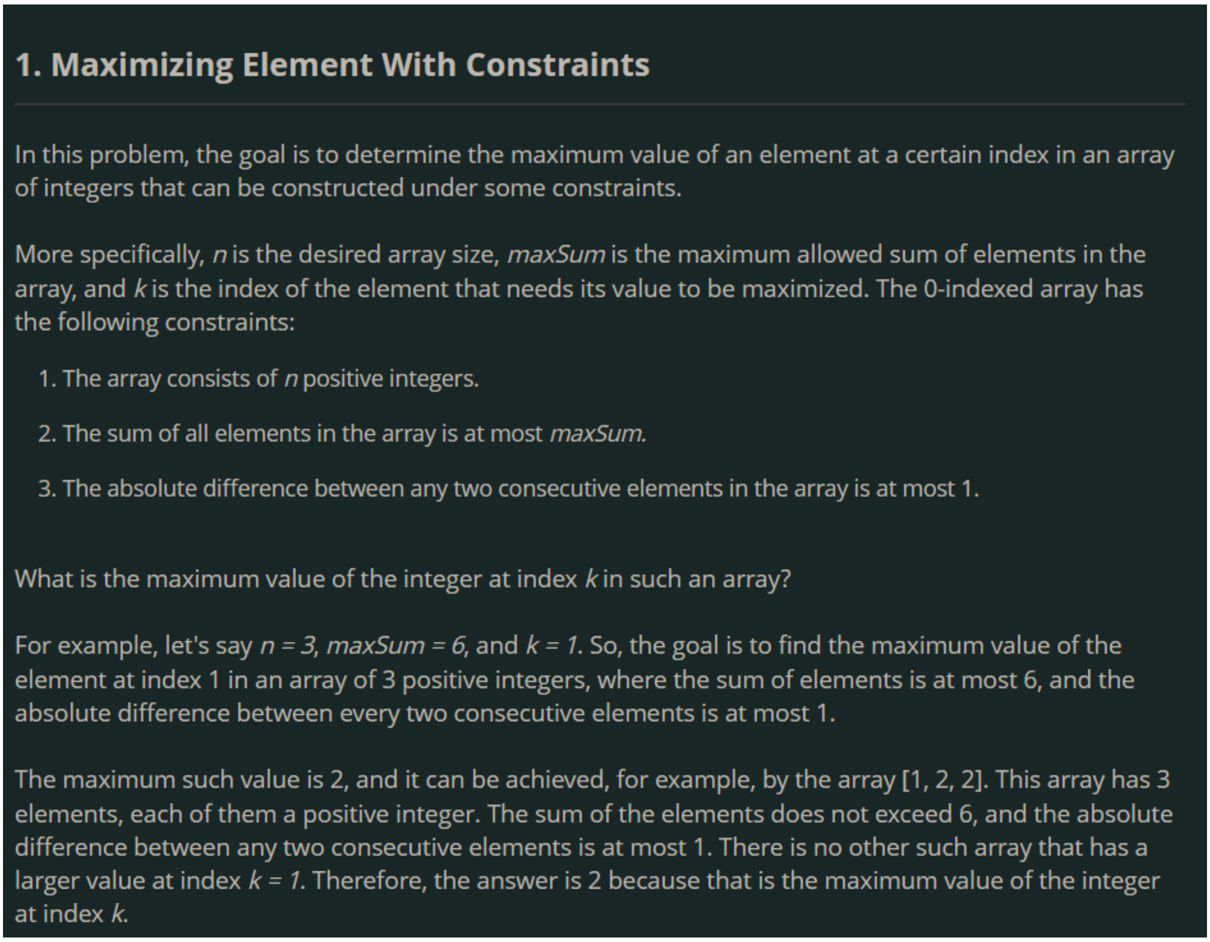


1. Maximizing Element With Constraints

In this problem, the goal is to determine the maximum value of an element at a certain index in an array of integers that can be constructed under some constraints.

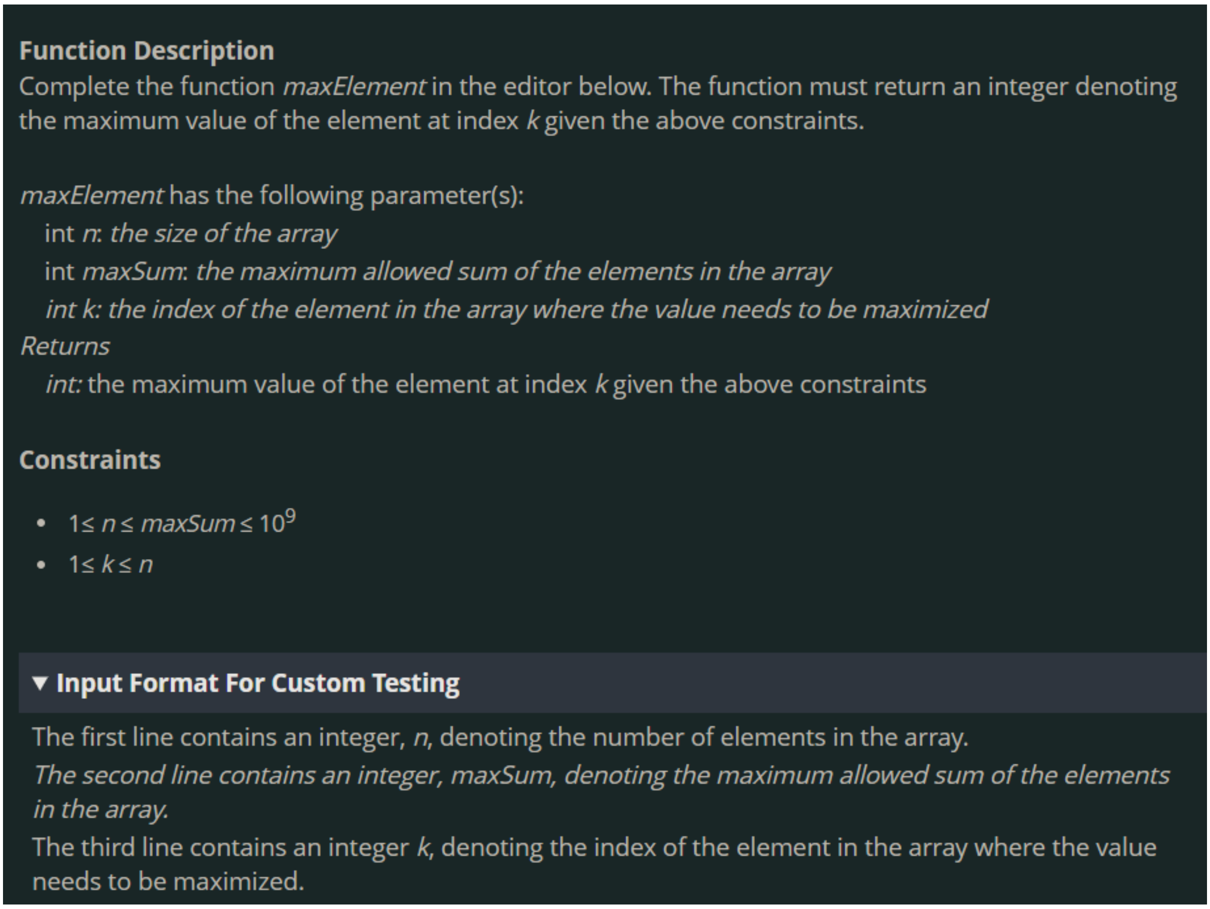
More specifically, *n* is the desired array size, *maxSum* is the maximum allowed sum of elements in the array, and *k* is the index of the element that needs its value to be maximized. The 0-indexed array has the following constraints:

1. The array consists of *n* positive integers.
2. The sum of all elements in the array is at most *maxSum.*
3. The absolute difference between any two consecutive elements in the array is at most 1.

What is the maximum value of the integer at index *k* in such an array?

For example, let's say *n = 3*, *maxSum = 6*, and *k = 1.* So, the goal is to find the maximum value of the element at index 1 in an array of 3 positive integers, where the sum of elements is at most 6, and the absolute difference between every two consecutive elements is at most 1.

The maximum such value is 2, and it can be achieved, for example, by the array [1, 2, 2]. This array has 3 elements, each of them a positive integer. The sum of the elements does not exceed 6, and the absolute difference between any two consecutive elements is at most 1. There is no other such array that has a larger value at index *k = 1.* Therefore, the answer is 2 because that is the maximum value of the integer at index *k.*



Function Description

Complete the function *maxElement* in the editor below. The function must return an integer denoting the maximum value of the element at index *k* given the above constraints.

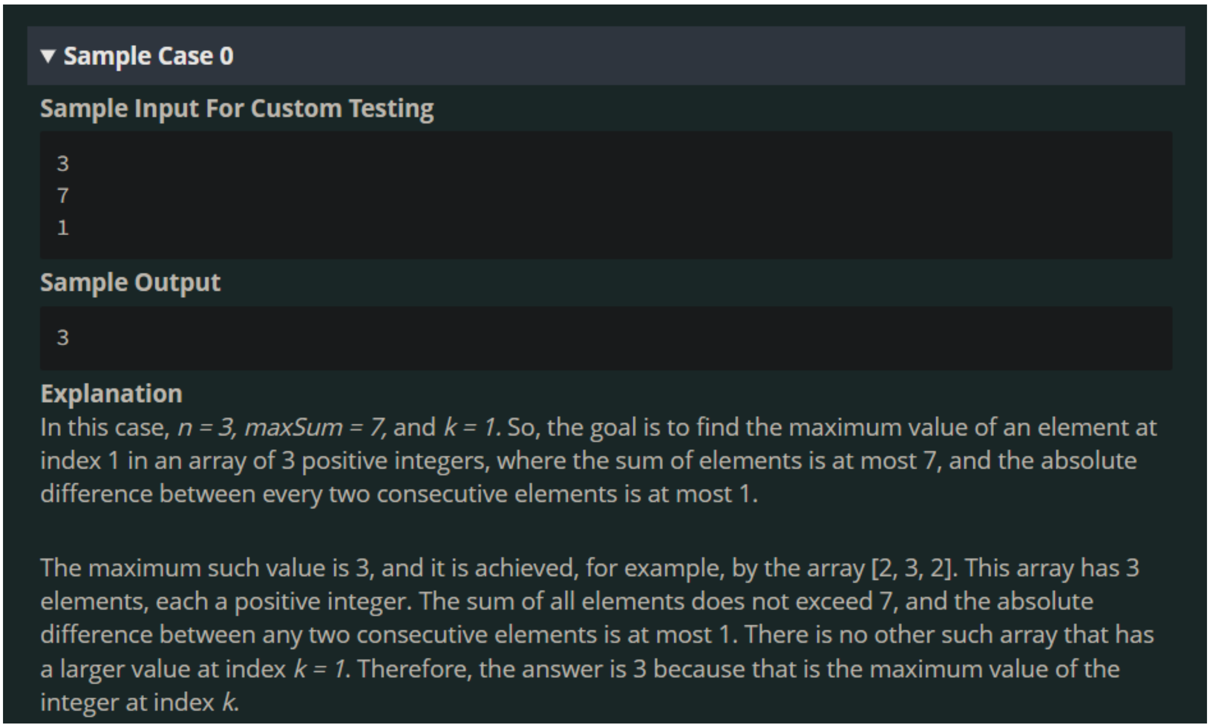
maxElement has the following parameter(s):
int *n: the size of the array*
int *maxSum: the maximum allowed sum of the elements in the array*
int k: the index of the element in the array where the value needs to be maximized
Returns
int: the maximum value of the element at index *k* given the above constraints

Constraints

- 1≤ *n* ≤ *maxSum* ≤ 10^9
- 1≤ *k* ≤ *n*

▼ Input Format For Custom Testing

The first line contains an integer, *n*, denoting the number of elements in the array.
The second line contains an integer, maxSum, denoting the maximum allowed sum of the elements in the array.
The third line contains an integer *k*, denoting the index of the element in the array where the value needs to be maximized.



▼ **Sample Case 0**

Sample Input For Custom Testing

```
3
7
1
```

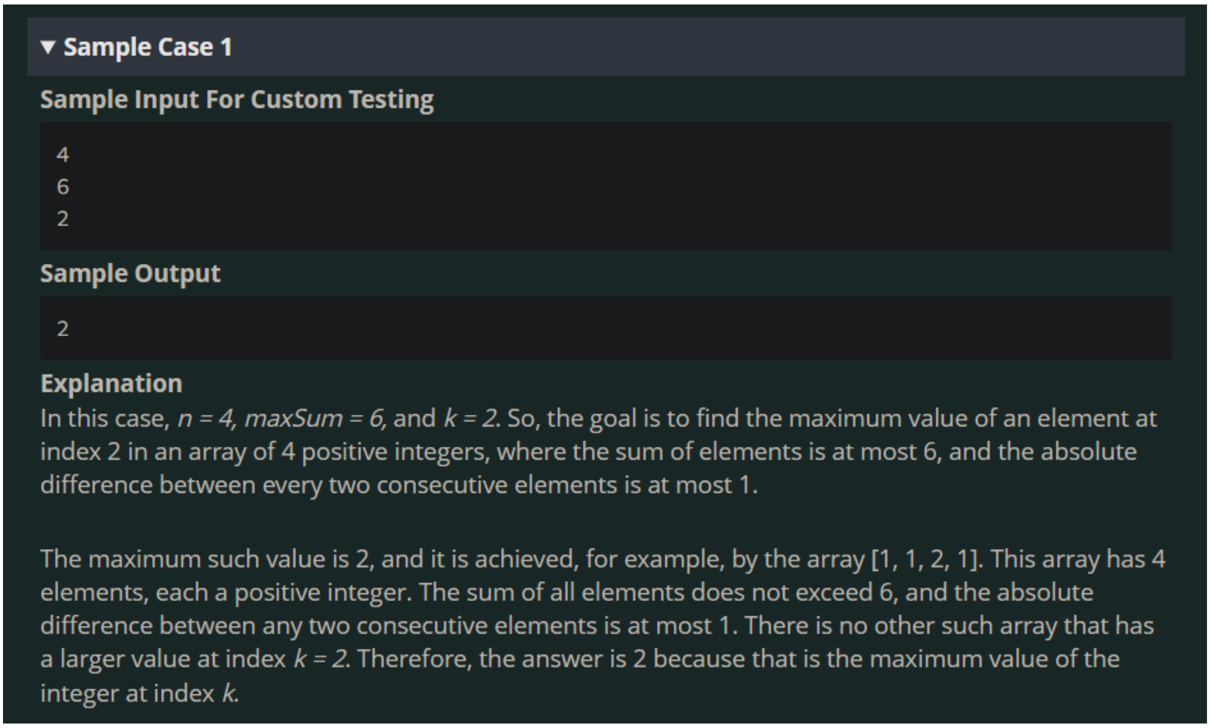
Sample Output

```
3
```

Explanation

In this case, *n = 3, maxSum = 7,* and *k = 1.* So, the goal is to find the maximum value of an element at index 1 in an array of 3 positive integers, where the sum of elements is at most 7, and the absolute difference between every two consecutive elements is at most 1.

The maximum such value is 3, and it is achieved, for example, by the array [2, 3, 2]. This array has 3 elements, each a positive integer. The sum of all elements does not exceed 7, and the absolute difference between any two consecutive elements is at most 1. There is no other such array that has a larger value at index *k = 1.* Therefore, the answer is 3 because that is the maximum value of the integer at index *k.*



▼ **Sample Case 1**

Sample Input For Custom Testing

```
4
6
2
```

Sample Output

```
2
```

Explanation

In this case, *n = 4, maxSum = 6,* and *k = 2.* So, the goal is to find the maximum value of an element at index 2 in an array of 4 positive integers, where the sum of elements is at most 6, and the absolute difference between every two consecutive elements is at most 1.

The maximum such value is 2, and it is achieved, for example, by the array [1, 1, 2, 1]. This array has 4 elements, each a positive integer. The sum of all elements does not exceed 6, and the absolute difference between any two consecutive elements is at most 1. There is no other such array that has a larger value at index *k = 2.* Therefore, the answer is 2 because that is the maximum value of the integer at index *k.*