Smallest integer

`Write a function solution(A)`

that, given an array A of N integers, returns the smallest positive integer (greater than 0) that does not occur in A.

For example, given A = [1, 3, 6, 4, 1, 2], the function should return 5.

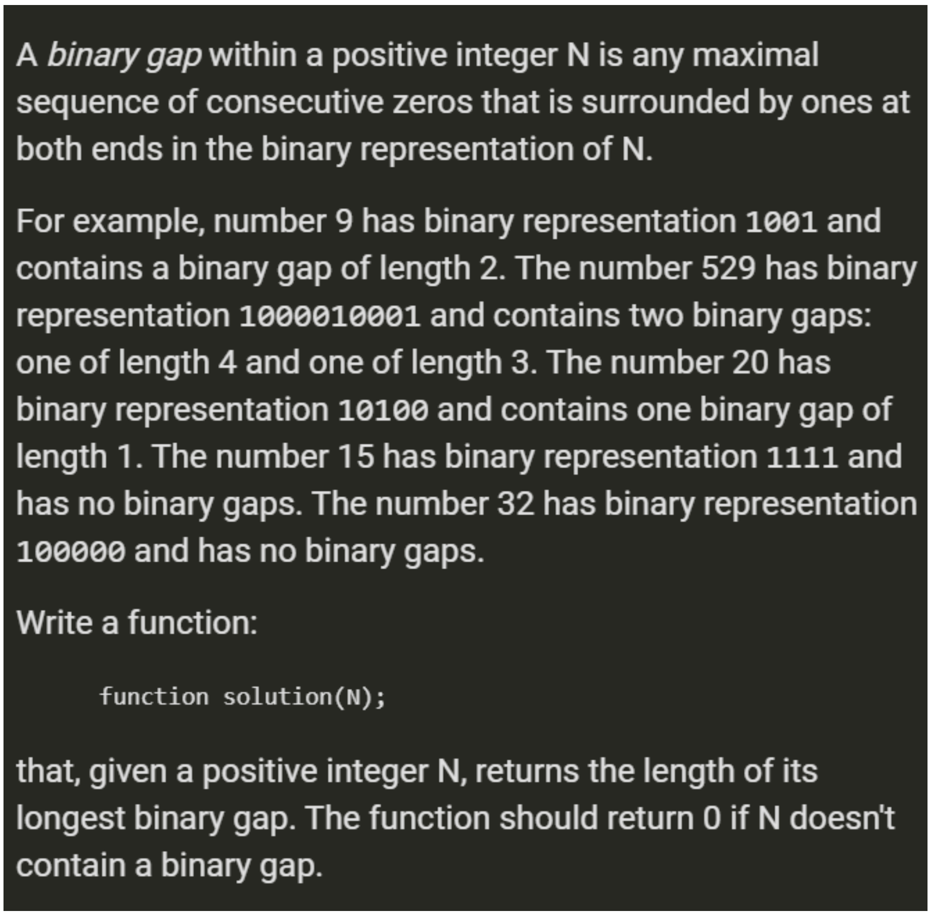
```
Given A = [1, 2, 3], the function should return 4.

Given A = [−1, −3], the function should return 1.
```

Assume that:

N is an integer within the range [1..100,000]; each element of array A is an integer within the range [−1,000,000..1,000,000]. Complexity:

BinaryGap



A *binary gap* within a positive integer N is any maximal sequence of consecutive zeros that is surrounded by ones at both ends in the binary representation of N.

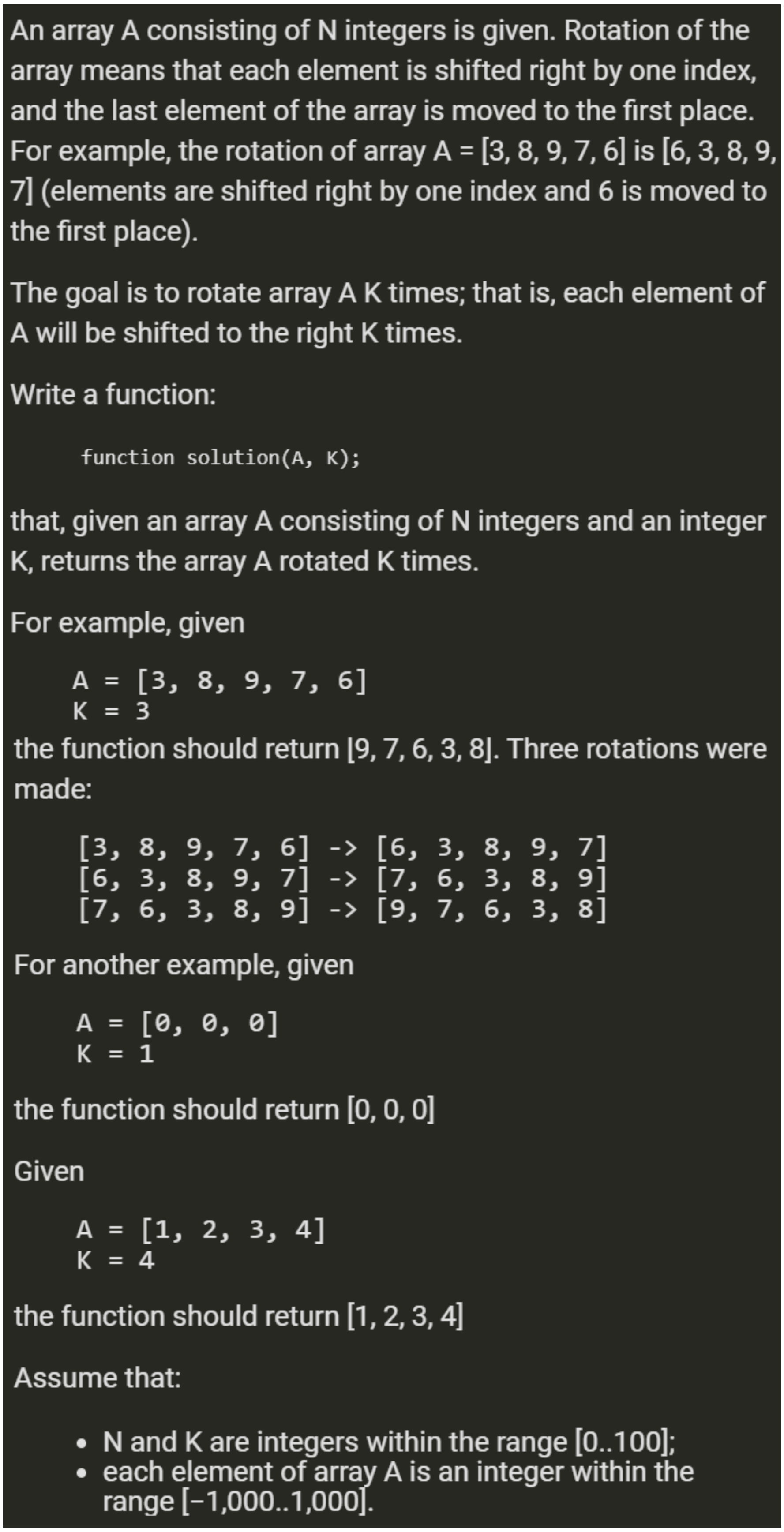
For example, number 9 has binary representation `1001` and contains a binary gap of length 2. The number 529 has binary representation `1000010001` and contains two binary gaps: one of length 4 and one of length 3. The number 20 has binary representation `10100` and contains one binary gap of length 1. The number 15 has binary representation `1111` and has no binary gaps. The number 32 has binary representation `100000` and has no binary gaps.

Write a function:

```
    function solution(N);
```

that, given a positive integer N, returns the length of its longest binary gap. The function should return 0 if N doesn't contain a binary gap.

CyclicRotation



An array A consisting of N integers is given. Rotation of the array means that each element is shifted right by one index, and the last element of the array is moved to the first place. For example, the rotation of array A = [3, 8, 9, 7, 6] is [6, 3, 8, 9, 7] (elements are shifted right by one index and 6 is moved to the first place).

The goal is to rotate array A K times; that is, each element of A will be shifted to the right K times.

Write a function:

```
function solution(A, K);
```

that, given an array A consisting of N integers and an integer K, returns the array A rotated K times.

For example, given

```
A = [3, 8, 9, 7, 6]
K = 3
```

the function should return [9, 7, 6, 3, 8]. Three rotations were made:

```
[3, 8, 9, 7, 6] -> [6, 3, 8, 9, 7]
[6, 3, 8, 9, 7] -> [7, 6, 3, 8, 9]
[7, 6, 3, 8, 9] -> [9, 7, 6, 3, 8]
```

For another example, given

```
A = [0, 0, 0]
K = 1
```

the function should return [0, 0, 0]

Given

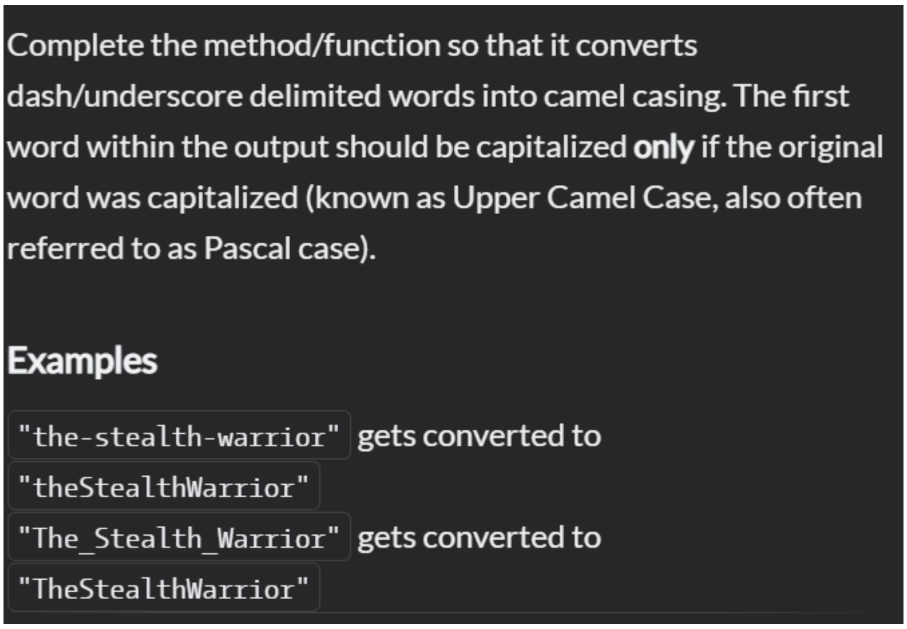
```
A = [1, 2, 3, 4]
K = 4
```

the function should return [1, 2, 3, 4]

Assume that:

- N and K are integers within the range [0..100];
- each element of array A is an integer within the range [−1,000..1,000].

Convert string to camel case



Complete the method/function so that it converts dash/underscore delimited words into camel casing. The first word within the output should be capitalized **only** if the original word was capitalized (known as Upper Camel Case, also often referred to as Pascal case).

Examples

`"the-stealth-warrior"` gets converted to `"theStealthWarrior"`
`"The_Stealth_Warrior"` gets converted to `"TheStealthWarrior"`

Find Words That Can Be Formed by Characters

You are given an array of strings `words` and a string `chars`.

A string is **good** if it can be formed by characters from chars (each character can only be used once).

Return *the sum of lengths of all good strings in words.*

Example 1:

```
Input: words = ["cat","bt","hat","tree"], chars = "atach"
Output: 6
Explanation: The strings that can be formed are "cat" and
"hat" so the answer is 3 + 3 = 6.
```

Example 2:

```
Input: words = ["hello","world","leetcode"], chars =
"welldonehoneyr"
Output: 10
Explanation: The strings that can be formed are "hello" and
"world" so the answer is 5 + 5 = 10.
```

Constraints:

- `1 <= words.length <= 1000`
- `1 <= words[i].length, chars.length <= 100`
- `words[i]` and `chars` consist of lowercase English letters.

ParkingBill

You parked your car in a parking lot and want to compute the total cost of the ticket. The billing rules are as follows:

- The entrance fee of the car parking lot is 2;
- The first full or partial hour costs 3;
- Each successive full or partial hour (after the first) costs 4.

You entered the car parking lot at time E and left at time L. In this task, times are represented as strings in the format `"HH:MM"` (where `"HH"` is a two-digit number between 0 and 23, which stands for hours, and `"MM"` is a two-digit number between 0 and 59, which stands for minutes).

Write a function:

```
function solution(E, L);
```

that, given strings E and L specifying points in time in the format `"HH:MM"`, returns the total cost of the parking bill from your entry at time E to your exit at time L. You can assume that E describes a time before L on the same day.

For example, given `"10:00"` and `"13:21"` your function should return 17, because the entrance fee equals 2, the first hour costs 3 and there are two more full hours and part of a further hour, so the total cost is 2 + 3 + (3 * 4) = 17. Given `"09:42"` and `"11:42"` your function should return 9, because the entrance fee equals 2, the first hour costs 3 and the second hour costs 4, so the total cost is 2 + 3 + 4 = 9.

Assume that:

- strings E and L follow the format "`HH:MM`" strictly;
- string E describes a time before L on the same day.

OddOccurrencesInArray

A non-empty array A consisting of N integers is given. The array contains an odd number of elements, and each element of the array can be paired with another element that has the same value, except for one element that is left unpaired.

For example, in array A such that:

```
A[0] = 9  A[1] = 3  A[2] = 9
A[3] = 3  A[4] = 9  A[5] = 7
A[6] = 9
```

- the elements at indexes 0 and 2 have value 9,
- the elements at indexes 1 and 3 have value 3,
- the elements at indexes 4 and 6 have value 9,
- the element at index 5 has value 7 and is unpaired.

Write a function:

```
function solution(A);
```

that, given an array A consisting of N integers fulfilling the above conditions, returns the value of the unpaired element.

For example, given array A such that:

```
A[0] = 9  A[1] = 3  A[2] = 9
A[3] = 3  A[4] = 9  A[5] = 7
A[6] = 9
```

the function should return 7, as explained in the example above.