You've just started to study impartial games, and came across an interesting theory. The theory is quite complicated, but it can be narrowed down to the following statements: solutions to all such games can be found with the *mex* function. *Mex* is an abbreviation of *minimum excludant*: for the given set s it finds the minimum non-negative integer that is not present in s.

You don't yet know how to implement such a function efficiently, so would like to create a simplified version. For the given set s and given an upperBound, implement a function that will find its *mex* if it's smaller than upperBound or return upperBound instead.

*Hint: for loops also have an else clause which executes when the loop completes normally, i.e. without encountering any breaks*

Example

* For s = [0, 4, 2, 3, 1, 7] and upperBound = 10,  
  the output should be  
  mexFunction(s, upperBound) = 5.

5 is the smallest non-negative integer that is not present in s, and it is smaller than upperBound.

* For s = [0, 4, 2, 3, 1, 7] and upperBound = 3,  
  the output should be  
  mexFunction(s, upperBound) = 3.

The minimum excludant for the given set is 5, but it's greater than upperBound, so the output should be 3.

Input/Output

* **[execution time limit] 4 seconds (py)**
* **[input] array.integer s**

Array of distinct non-negative integers.

*Guaranteed constraints:*  
0 ≤ s.length ≤ 100,  
0 ≤ s[i] ≤ 100.

* **[input] integer upperBound**

A positive integer.

*Guaranteed constraints:*  
1 ≤ upperBound ≤ 100.

* **[output] integer***Mex* of s if it's smaller than upperBound, or upperBound instead.
* def mexFunction(s, upperBound):
* found = -1
* for i in range(upperBound):
* if not i in s:
* found = i
* break
* else:
* return upperBound
* return found