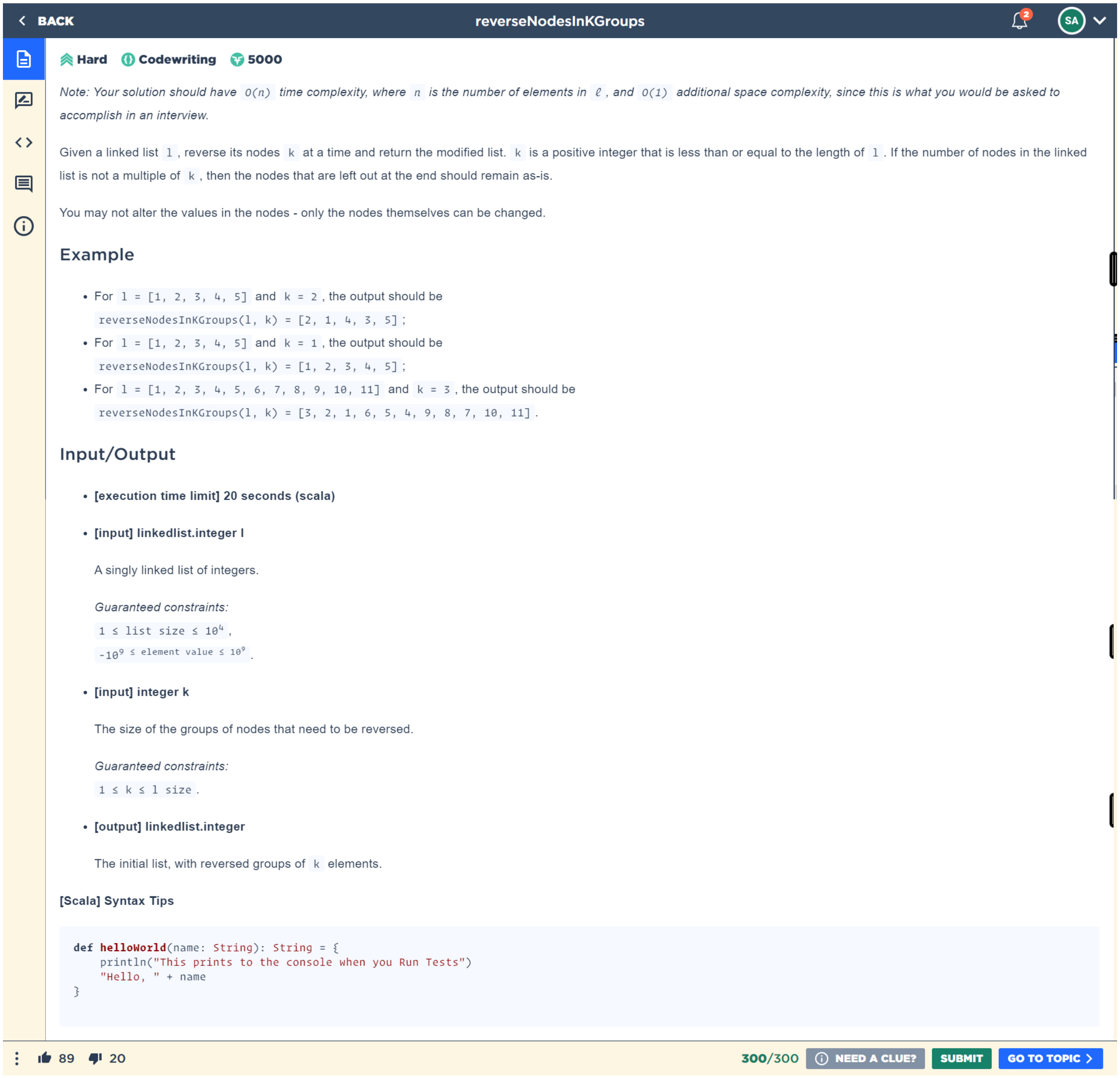

Note: Your solution should have `O(n)` time complexity, where `n` is the number of elements in `ℓ`, and `O(1)` additional space complexity, since this is what you would be asked to accomplish in an interview.

Given a linked list `l`, reverse its nodes `k` at a time and return the modified list. `k` is a positive integer that is less than or equal to the length of `l`. If the number of nodes in the linked list is not a multiple of `k`, then the nodes that are left out at the end should remain as-is.

You may not alter the values in the nodes - only the nodes themselves can be changed.

Example

- For `l = [1, 2, 3, 4, 5]` and `k = 2`, the output should be `reverseNodesInKGroups(l, k) = [2, 1, 4, 3, 5]`;
- For `l = [1, 2, 3, 4, 5]` and `k = 1`, the output should be `reverseNodesInKGroups(l, k) = [1, 2, 3, 4, 5]`;
- For `l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]` and `k = 3`, the output should be `reverseNodesInKGroups(l, k) = [3, 2, 1, 6, 5, 4, 9, 8, 7, 10, 11]`.

Input/Output

- **[execution time limit] 20 seconds (scala)**

- **[input] linkedlist.integer l**

  A singly linked list of integers.

  Guaranteed constraints:
  $1 \le \text{list size} \le 10^4$,
  $-10^9 \le \text{element value} \le 10^9$.

- **[input] integer k**

  The size of the groups of nodes that need to be reversed.

  Guaranteed constraints:
  `1 ≤ k ≤ l size`.

- **[output] linkedlist.integer**

  The initial list, with reversed groups of `k` elements.

[Scala] Syntax Tips

```
def helloWorld(name: String): String = {
    println("This prints to the console when you Run Tests")
    "Hello, " + name
}
```