99 problems in Haskell - Part 1 (Lists)

1. **myLast :: [a] → a**
   
   Find the last element of a list. Assume the list is non empty.

2. **myButLast :: [a] → a**
   
   Find the last but one element of a list. Assume the list has, at least, two elements.

3. **elementAt :: Int → [a] → a**
   
   Find the \(k\)-th element of a list. The first element in the list is number 1. Assume the list has, at least, \(k\) elements.

4. **myLength :: [a] → Int**
   
   Find the number of elements of a list.

5. **myReverse :: [a] → [a]**
   
   Reverse a list.

6. **isPalindrome :: (Eq a) ⇒ [a] → Bool**
   
   Find out whether a list is a palindrome.

7. **myFlatten :: [[a]] → [a]**
   
   Flatten a two-level nested list structure.

8. **compress :: (Eq a) ⇒ [a] → [a]**
   
   Eliminate consecutive duplicates of list elements.

9. **pack :: (Eq a) ⇒ [a] → [[a]]**
   
   Pack consecutive duplicates of list elements into sublists. If a list contains repeated elements they should be placed in separate sublists.

10. **encode :: (Eq a) ⇒ [a] → [(Int, a)]**
    
    Run-length encoding of a list. Consecutive duplicates of elements are encoded as lists \((n, e)\) where \(n\) is the number of duplicates of the element \(e\).

**Scoring**

Each item scores 10 points.

**Sample input**

<table>
<thead>
<tr>
<th>Function</th>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>myLast</td>
<td>[1..5]</td>
<td>5</td>
</tr>
<tr>
<td>myLast</td>
<td>&quot;hello&quot;</td>
<td>&quot;hello&quot;</td>
</tr>
<tr>
<td>myButLast</td>
<td>[1..5]</td>
<td>4</td>
</tr>
<tr>
<td>elementAt</td>
<td>3 [1..5]</td>
<td>3</td>
</tr>
<tr>
<td>myLength</td>
<td>[1..5]</td>
<td>5</td>
</tr>
<tr>
<td>myReverse</td>
<td>[1..5]</td>
<td>[5..1]</td>
</tr>
<tr>
<td>isPalindrome</td>
<td>&quot;madam&quot;</td>
<td>True</td>
</tr>
<tr>
<td>myFlatten</td>
<td>[[1..5],[3..4],[2..4]]</td>
<td>[1..5],[3..4],[2..4]</td>
</tr>
<tr>
<td>compress</td>
<td>&quot;aaacaabb&quot;</td>
<td>&quot;aaacaabb&quot;</td>
</tr>
<tr>
<td>pack</td>
<td>&quot;aaacaabb&quot;</td>
<td>[[1..3],[1..3],[3..1]]</td>
</tr>
<tr>
<td>encode</td>
<td>&quot;aaacaabb&quot;</td>
<td>[(3, 'a'), (3, 'a'), (1, 'b')]</td>
</tr>
</tbody>
</table>
Sample output

5
'o'
4
3
5

[5, 4, 3, 2, 1]
True
[1, 2, 3, 4, 5, 3, 4, 2, 3, 4]
"acab"
["aaa", "c", "aa", "bb"]
[(3, 'a'), (1, 'c'), (2, 'a'), (2, 'b')]