Exercise 3.1 Bite-sized Haskell Tasks

a) `Swap the values of the first and last element in a list`

```
2  swap_first_and_last :: [a] -> [a]
3  swap_first_and_last [] = []
4  swap_first_and_last [x] = [x]
5  swap_first_and_last xs = last xs : tail (init xs) ++ [head xs]
```

First check for the case that the list is empty, if so, return it. Then check the case if the list contains only a single item, if so, again, return it. Only in the third case, when the list contains 2 or more elements, swap the first and last.
`last xs` = last element of xs
`tail (init xs)` = all elements of xs without the first and last
`head xs` = first element of xs

b) `Return, as a Boolean, whether two consecutive elements in a list are the same`

```
8  contains_equal_consec :: Eq a => [a] -> Bool
9  contains_equal_consec [] = False
10 contains_equal_consec [_] = False
11 contains_equal_consec (x:y:xs) = x == y || contains_equal_consec(y:xs)
```

Exercise 3.2 Classes and Inheritance

Exercise 3.3 Operator Overloading

Exercise 3.4 Templates

Exercise 5 and 6 Tic Tac Toe, The Ultimate Game