Till Mossakowski

Formal Methods for Software Development
Exercise sheet 2

Exercise 1:

Consider the following implementation of the `sublist` function:

```
sublist :: Eq a => [a] -> [a] -> Bool

sublist [] _ = True
sublist _ [] = False
sublist (x:xs) (y:ys) | x==y = sublist xs ys
                      | otherwise = sublist (x:xs) ys
```

and let this be the intended one. Consider the following non-intended implementations:

```
sublist [] _ = True
sublist _ [] = False
sublist (x:xs) (y:ys) | x==y && xs == take (length xs) ys = True
                      | otherwise = sublist (x:xs) ys

sublist l1 l2 = all (‘elem‘ l2) l1

sublist l1 l2 = all (‘elem‘ l1) l2
```

Write a set of QuickCheck specifications that only passes for the intended implementation. Try to formulate general quantified properties rather than individual test cases.

[Hint: Use `import Debug.QuickCheck` to import the QuickQueck module]

Exercise 2:

Extend the monadic implementation of queues with additional operation(s), e.g. for taking the rear of a queue. Formulate additional QuickCheck properties characterizing the new operations.