Till Mossakowski

Formal Methods for Software Development
Exercise sheet 1

Exercise 1:

Write a set of HUnit test cases for a Haskell program that tests if a list is a sublist of another list.

[Hints: Use the function `error :: String -> a` to generate a dummy implementation that is needed for writing the test cases. Use auxiliary functions to abstract from common patterns in the test cases.]

Then implement the program, and test it.

[Hint: Use `elem :: (Eq a) => a -> [a] -> Bool` to test membership, and `all :: (a -> Bool) -> [a] -> Bool` to test whether a predicate is satisfied by all members of a list. Haskell supports the so-called section notation: `(`elem` l)` is a test for membership in the list `l`.]

Discuss possible variants of the sublist function.

Exercise 2:

Write a set of HUnit test cases for a Haskell program that computes the difference of two lists. [Hints as above]

Then implement the program, and test it.

[Hint: Use `filter :: (a -> Bool) -> [a] -> [a]` to obtains those elements of a list satisfying a given predicate. `not :: Bool -> Bool` is negation of Booleans, and `(.) :: (b -> c) -> (a -> b) -> a -> c` is function composition.]