COMP 3649 Exercise Set 3

Given:	Thursday, February 6
Complete by:	Wednesday, February 12
Quiz:	Thursday, February 13 (in-class at 8:30 a.m.)

For question 1, attempt to develop written answers by yourself. Once you have done your best, you may discuss your answers with two-to-three other students. Then, by yourself (without looking at notes from the discussion), revise your own written answers as you believe appropriate.

1. The following polymorphic functions are all pre-defined in Haskell's standard prelude:

   `fst`, `snd`, `head`, `tail`, `take`, `drop`, `unzip`, `concat`

   Write (declare and define) your own version of each. Use the same names and make use of "`import Prelude hiding (…)`"

   Recall: You can use the "`:i`" and "`:t`" GHCi commands to query information, such as type, for each. Be sure to try out example invocations of each prelude version before defining your own.

For questions 2-5, attempt to develop written answers by yourself. Once you have done your best, you may discuss your answers with two-to-three other students. Then, by yourself (without looking at notes from the discussion), revise your own written answers as you believe appropriate.

2. Write (declare and define) your own O(*n*) version of the standard prelude `reverse` function. A naïve implementation that uses append (`++`) to repeatedly place a head element of a given list at the end of a new list is O(n^2) – why? Hint: your version should make use of a helper function that takes a *pair* of lists and repeatedly "shunts" the headmost element of the first list to the front of the second list.

3. Write (declare and define) a function that takes a list of `Either a b` values and produces a list of just the `a` values, preserving their order.

4. Write (declare and define) a function that takes a list of integers, as well as a second argument of type integer, and produces a list of pairs of quotients and remainders obtained by integer-dividing the second argument by each list element. Utilize `Maybe` to deal with the case of division by zero.

5. Write (declare and define) a function that takes a list of integers and returns the maximum value. For the return type, utilize `Maybe` to handle the case in which the result is undefined.

Develop complete answers to questions 6-8 working in a group of two-to-four students (but ensure you can recreate correct answers on your own).

6. Define a polymorphic data type for non-empty lists (i.e., there is no empty list value for this type).

7. Define a polymorphic tree data type such that each node contains a value as well as either two or three subtrees. Trees may also be empty.

8. Based on your answer to question 7, write (declare and define) a function that counts the number of elements in a given tree.

For questions 9-10, develop written answers yourself without conferring or checking your work with other students.

9. Define a polymorphic tree data type such that each node contains a value as well as a non-empty list of subtrees. Note that two distinct nodes in a given tree may have different numbers of subtrees. Trees may also be empty.

10. Based on your answer to question 9, write (declare and define) a function that counts the number of elements in a given tree.