HW 1 Solution: Haskell

CS 440: Programming Languages and Translators
Was due Sat Feb 2, 11:59 pm

1. \texttt{last'} x = \texttt{head (reverse x)}

2. \texttt{head\_repeats n x = (take n x) == (take n (drop n x))}

3. The definition of \texttt{f} fails because the it's inconsistent about how many arguments \texttt{f} takes. One fix is to make \texttt{f} always take two arguments but have the first case ignore the second argument. The new \texttt{f \::\:: (Num a, Eq a) => a \rightarrow a.}

\begin{verbatim}
  f 0 _ = 1    # (You can have a variable instead of _)
  f n x = n * n * x
\end{verbatim}

Another fix is to use a list as the parameter. Note this version of \texttt{f} will get a runtime error if you give it a list of length > 2, the empty list, or a singleton list that doesn't contain 0.

\begin{verbatim}
  f [0] = 1
  f [n, x] = n * n * x
\end{verbatim}

(I'm sure there are other ways to fix this error.)

4. Here's one definition.

\begin{verbatim}
  swap\_ends [] = []
  swap\_ends [y] = [y]
  swap\_ends x
      = last x : (reverse (drop 1 (reverse (drop 1 x))))
            ++ [head x]
\end{verbatim}

5. The pattern on the lhs is illegal: You can't build lists using \texttt{++}. You can build lists of an explicit length using square brackets (for example, 3 elements in \texttt{[e_1, e_2, e_3]}) and you can also build non-empty lists using colon: (for example, \texttt{e_1 : e_2}, where \texttt{e_2} is a list of unspecified length). We can't rewrite the lhs pattern because we can only access the last element of the list if we know how long the list is, and we can only write a finite number of cases:

\begin{verbatim}
  swap\_ends [] = []
  swap\_ends [x] = [x]
\end{verbatim}
\[
\text{swap\_ends} \ [x, z] = [z, x] \\
\text{swap\_ends} \ [x, y_1, z] = [z, y_1, x] \\
\text{swap\_ends} \ [x, y_1, y_2, z] = [z, y_1, y_2, x] \\
\# \text{ and so on}
\]

(Also note the patterns above don't define a sublist \(y\); you have to name each element of \(y\) individually.)

6. The problem is that \(x\) is of typeclass \texttt{Num}, which doesn't support \(/\). (We can verify this using \texttt{:info Num in ghci}). We need \texttt{Fractional} instead (check out \texttt{:info Fractional}). The correct declaration is

\[
f :: (\text{Fractional} \ a) \Rightarrow a \Rightarrow [a] \\
f \ x = [x \ / \ 2]
\]