Fill in the type of each of the following Haskell expressions. You may assume that the only numeric type is \texttt{Int} (this is not true in Haskell, of course, but will suffice for our practice now). So we would write:

\begin{verbatim}
f :: Int -> Int  
f x = x * 3
\end{verbatim}

Concrete types, type variables, and currying.

\begin{verbatim}
x ::
x = if 3 > 4 then 15 else 40
y ::
y = [4,5,6,7]
z ::
z = [[4,5], [1,10,0,0], [5]]
\end{verbatim}

These are higher-order functions. Use parentheses in your types to make it clear how to group the ->.

\begin{verbatim}
applyToThree ::
applyToThree f = f 3
apply ::
apply f x = f x
makeAdder ::
makeAdder x = \y -> x + y
\end{verbatim}

Classic list HOFs.

\begin{verbatim}
map ::
map f [] = []
map f (x:xs) = (f x):(map f xs)
filter ::
filter f [] = []
filter f (x:xs) =  
  if f x  
  then  
    x:(filter f xs)  
  else 
    filter f xs
foldl ::
foldl f i [] = i
foldl f i (x:xs) = 
  let z = f i x  
    in foldl f z xs
\end{verbatim}