HASKELL AND CURRYING

Problem 1. We have seen the colon operator used in list functions; e.g.:

```haskell
sumList [] = 0
sumList (x:xs) =
    x + sumList xs
```

Remember that we can convert operators to functions by wrapping them with parentheses.

What is...

```
:t (:
```

Problem 2. We can create functions with lambda, which is denoted by the backslash. E.g., here is a function that will add two numbers (bound to `add2`):

```haskell
add2 = \ a b -> a + b
```

What is `:t add2`?

Problem 3. Using lambda, write a function `add1` which increments its argument.

Problem 4. Write a function `gt6`, which returns the Boolean `True` if its parameter is greater than 6, or `False` otherwise. Use any method you prefer.
Problem 5. Write a Scheme function named \texttt{curry} which accepts \texttt{fcn}, a function of two parameters, and \texttt{param1}, its first parameter. This function should produce a new function which accepts \texttt{fcn}'s second parameter and then applies it to both of them.

E.g.:

\[
((\text{curry } + 3) 4) \Rightarrow 7
\]

\[
\text{(define (curry fcn param1)}
\]

Problem 6. Draw the environment diagram that results from

\[
\text{(define curried (curry + 3))}
\]