Announcements

- HW 6 is due Friday after break
- Today is in-class work day
Haskell

Purely functional — NO variables.
- Strongly typed (uses inference)
Functions (p. 2)

Example

Notes: It is considered good practice to type your functions.

\[\text{function name :: type } \rightarrow \text{ type}\]

Type could be list, too —

\[
\begin{align*}
\text{go check :} & : t \text{ "Hello" : t } (\text{"a", } \text{"b", } \text{"c"}) \\
& : t \text{ } \text{"a"} \\
& : t \text{ } (\text{True, } \text{"a"})
\end{align*}
\]
Ex: \texttt{remove NonUpperCase} :: [\texttt{char}] \rightarrow [\texttt{char}]

\begin{align*}
\text{remove NonUpperCase} \quad &\quad \texttt{st} = \\
&\quad [c \mid c \leftarrow \texttt{st}, c \text{ 'elem' } [\texttt{A}..\texttt{Z}]]
\end{align*}

Ex: \texttt{addThree} :: \texttt{Int} \rightarrow \texttt{Int} \rightarrow \texttt{Int} \rightarrow \texttt{Int} \\
\text{addThree } x \ y \ z = x + y + z
Nice Feature:

If you don't know the type, you can write your function and then check it!
Type Variables

Try : t head
(What is head again?)

\xrightarrow{\text{returns first element}}

Haskell allows for polymorphism in this way:

Try \texttt{fst} - takes tuple first element
Exercise

Get into groups (preferably w/ at least 1 computer per group).

Write the fibonacci function in Haskell.
(Pl ease type it).

Note: Submit to me w/ names of group for credit!
\textbf{fib} :: \texttt{Integer} \to \texttt{Integer}

\begin{align*}
\text{fib} \ 0 &= 0 \\
\text{fib} \ 1 &= 1 \\
\text{fib} \ n &= \text{fib} \ (n-1) + \text{fib} \ (n-2)
\end{align*}
Type Classes

Not an O-O class!
An interface that defines some behavior.
(More like Java interfaces.)

: t (==)

What is ⇒?
  a class constraint
Type classes

This means that the type of the two values (a) must be a member of the Eq class.

**Eq**: for types that support equality testing

**Ord**: types that have an ordering

**Show**: types that can be presented as strings

Read:  → (opposite)
Strong pattern Matching

Can use pattern matching in unexpected ways.

Add vectors example

Another:

```latex
> \text{let } my\text{\_list } = [(1, 3), (4, 3), (5, 6)]
> [(a+b) | (a, b) \in my\text{\_list}]
```
Our own head function

head' :: [a] -> a
head' [] = error 'Can't call head on empty list'
head' (x:_) = x

See first two.o.hs