Notes for Lecture 12 (Fall 2022 week 6, part 1): Polymorphism, continued

Jana Dunfield

October 16, 2022

Code for this lecture is in `lec11.hs` (there is no lec12.hs).

```
-- takes two arguments, puts them in a pair
--  (reversed: first argument becomes second component,
--   second component becomes first argument)
pair :: a -> b -> (b, a)
pair x y = (y, x)
```

1 mymap and pair

This file explains the use of `pair` with `mymap`.

```
*Lec11> :load lec11
[1 of 1] Compiling Lec11            ( lec11.hs, interpreted )
Ok, modules loaded: Lec11.
*Lec11> :type mymap
mymap :: (a -> b) -> [a] -> [b]
```


As with the original integers-only `mymap` (lec11.hs), we can use the polymorphic `mymap` on an integer list to create a new integer list:

```
*Lec11> mymap (\x -> x + 1) [100,200,300]
[101,201,301]
*Lec11> mymap (\x -> 1) [100,200,300]
[1,1,1]
```

We can also use `mymap` on an integer list, returning a Boolean list, by giving `mymap` the function `(\x -> False)`:

```
*Lec11> mymap (\x -> False) [100,200,300]
[False,False,False]
*Lec11> :type mymap (\x -> False) [100,200,300]
mymap (\x -> False) [100,200,300] :: [Bool]
```

Perhaps more usefully, we can give `mymap` the function `(\x -> x > 150)`, returning a list whose elements are `True` for integer elements are greater than `150`:

```
*Lec11> mymap (\x -> x > 150) [100,200,300]
[False,True,True]
```

```
*Lec11> :type pair
pair :: a -> b -> (b, a)
```

The `pair` function I defined takes two arguments and puts them in a pair, reversed:

```
*Lec11> (pair True 3)

(3,True)
```

The first argument to `mymap` needs to be a function that takes *one* argument. I wrote `(pair True)`, which is a function that takes one argument: `pair` takes two, so `(pair True)` is "waiting" for its second argument.

```
*Lec11> mymap (pair True) [100,200,300]
[(100,True),(200,True),(300,True)]
*Lec11>
```


The following code is equivalent:

```
*Lec11> mymap (\z -> pair True z) [100,200,300]
[(100,True),(200,True),(300,True)]
*Lec11>
```


For the rest of this lecture, move back to 360-lec11.pdf, from "You might have noticed that we've used various kinds of lists".