CS380: Modern Functional Programming
Prof. Richard Eisenberg
Spring 2017

Higher-Order Functions

Suppose the following functions have the types given:

$frob :: Int \to String \to Bool$
$wurble :: Bool \to Int$
$map :: (a \to b) \to [a] \to [b]$
$filter :: (a \to Bool) \to [a] \to [a]$
$zipWith :: (a \to b \to c) \to [a] \to [b] \to [c]$
$(\$) :: (a \to b) \to a \to b$

Give types to each of the following (all considered separately), or write that the definition is ill-typed:

1. $f\ x = x$

   $f ::$

2. $f\ x\ y = x\ y$

   $f ::$

3. $f\ x\ y = y\ x$

   $f ::$

4. $f\ (x : xs) = x$

   $f ::$

5. $f\ (x : xs) = xs$

   $f ::$

6. $f\ b = \textbf{if}\ b\ \textbf{then}\ b\ \textbf{else}\ b$

   $f ::$

7. $f\ x\ y\ z = x\ (y\ z)$

   $f ::$

8. $f = map\ wurble$

   $f ::$

9. $f = map\ frob$

   $f ::$

10. $f = filter\ wurble$

    $f ::$

11. $f = filter\ frob$

    $f ::$

12. $f = filter\ (wurble\ False)$

    $f ::$

13. $f = \mathit{filter}\ (\mathit{frob}\ 5)$
    $f ::$
14. $f = \mathit{zipWith}\ (\$)$
    $f ::$
15. $f\ x = \mathit{filter}\ ((\$)\ x)$
    $f ::$
16. $f\ x = x\ x$
    $f ::$

We now assume the following definitions:

$\mathit{id} :: a \to a$
$\mathit{id}\ x = x$
$\mathit{const} :: a \to b \to a$
$\mathit{const}\ a\ \_ = a$

Figure out what the following reduce to, or say that the expression is ill-typed:

17. $\mathit{map}\ \mathit{id}\ [1, 2, 3] \longrightarrow$
18. $\mathit{map}\ (\mathit{const}\ \mathit{False})\ [\texttt{'x'}, \texttt{'y'}, \texttt{'z'}] \longrightarrow$
19. $\mathit{filter}\ (\mathit{const}\ \mathit{False})\ \texttt{"abc"} \longrightarrow$
20. $\mathit{filter}\ \mathit{id}\ [\mathit{True}, \mathit{False}, \mathit{True}] \longrightarrow$
21. $\mathit{zipWith}\ \mathit{id}\ [\mathit{id}, \mathit{not}]\ [\mathit{False}, \mathit{True}] \longrightarrow$
22. $\mathit{id}\ \mathit{not}\ \mathit{True} \longrightarrow$
23. $\mathit{id}\ \mathit{id}\ \texttt{'x'} \longrightarrow$

Rewrite the following definitions into one-liners using *map* and *filter*:

24.
$$\begin{aligned} f\ [\,] &= [\,] \\ f\ (x : xs) &= x + 1 : f\ xs \end{aligned}$$

25.
$$\begin{aligned} &f\ [\,] = [\,] \\ &f\ (x : xs) \\ &\quad \mid \mathit{even}\ x = f\ xs \\ &\quad \mid \mathit{otherwise} = x : f\ xs \end{aligned}$$

26.
$$\begin{aligned} &f\ [\,] = [\,] \\ &f\ (x : xs) \\ &\quad \mid \mathit{even}\ x = x\ `\mathit{div}`\ 2 : f\ xs \\ &\quad \mid \mathit{otherwise} = f\ xs \end{aligned}$$

27.
```
f []             = []
f (x : xs)
   | even y      = y : f xs
   | otherwise = f xs
   where
     y = x `div` 2
```

Consider the following definitions:

```
[] ++       ys = ys
(x : xs) ++ ys = x : (xs ++ ys)
concatMap _ []       = []
concatMap f (x : xs) = f x ++ concatMap f xs
```

28. (++)::

29. *concatMap*::

30. Use *concatMap* to write a function *dup* that duplicates every element in a list. That is *dup* [`'x'`, `'y'`, `'z'`] evaluates to [`'x'`, `'x'`, `'y'`, `'y'`, `'z'`, `'z'`].