Chapter 3 Homework:

12/6/2010

Q: What are the types of the following values?

```
['a','b','c']
('a','b','c')
[(False,'0'),(True,'1')]
([False,True],['0','1'])
[tail,init,reverse]
```

A:

```
[Char]
(Char,Char,Char)
[(Bool,Char)]
([Bool],[Char])
[[a] → [a]]
```

Q: What are the types of the following functions?

```
second xs = head (tail xs)
swap (x,y) = (y,x)
pair x y = (x,y)
double x = x ∗ 2
palindrome xs = reverse xs == xs
twice f x = f (f x)
```

A:

```
second :: [a] → a
```

```
swap :: (a, b) → (b, a)
pair :: a → b → (a, b)
double :: Num a ⇒ a → a
palindrome :: [a] → Bool
twice :: (a → a) → a → a
```

Q: Check answers using Hugs (GHCI).

A: All answers checked and were correct except fore palindrome, where I forgot to put the typeclass constraint on the input. Revised answer as follows:

```
palindrome :: (Eq a) ⇒ [a] → Bool
```

Q: Why is it not feasible in general for function types to be instances of the Eq class? When is it feasible? Hint: two functions of the same type are equal if they always return equal results for equal arguments.

A: Functions cannot automatically be a part of the eq typeclass, because it would require checking the following:

- That all possible inputs were the same (Not just the same type).
- That all possible inputs were mapped into the same output.

Since possible inputs are often infinite, this would not only be unfeasible but impossible. Unless there is a simpler way to test for function equivalence that I am unaware of.