COMP105 Lecture 9

Where Syntax

New syntax: `where`

Sometimes its useful to bind names across a **whole function**

```
remove_twos [] = []
remove_twos (x:xs)
    | x == 2    = rest
    | otherwise = x : rest
    where rest = remove_twos xs
```

`where` syntax

`where` comes at the **end** of a function

```
initials first last = [f] ++ ". " ++ [l] ++ "."
    where (f:_) = first
          (l:_) = last
```

Like `let` it can bind any number of names

- These names are available throughout the function
- They are not variables, of course
- You can do pattern matching in `where` clauses (and also in `let` expressions too)

where vs let

`let` is an **expression**

```
ghci> (let a = 1 in a) + (let a = 2 in a)
3
```

`where` is **syntax**

- One `where` clause per function
- Particularly useful when used with guards

```
remove_twos (x:xs)
    | x == 2    = rest
    | otherwise = x : rest
    where rest = remove_twos xs
```

Exercises

1. Use `where` to write a function `func` that takes one argument `x` and computes `a = 2 * x + 1`, `b = a*a - x`, and returns `a + b + x`

2. Use `where` to write a recursive function `countTwos` that takes a list of integers and counts the number of twos that appear in the list