COMP105 Lecture 18

Higher Order Programming Example

Mark averages

We have a le of **student marks**

- For assignment 1, 2, 3, and the class test

aaaa	70	65	67	60
bbbb	55	60	55	65
cccc	40	40	40	40
dddd	80	60	75	60
cccc	0	0	0	100

Mark averages

We want to produce a file of **mark averages**

```
aaaa  65.5
bbbb  58.75
cccc  40.0
dddd  68.75
cccc  25.0
```

Reading files in Haskell

We can read a file using **readFile**

- ▶ This is an IO function
- ▶ We will study this in more detail later on

```
ghci > readFile "marks.csv"
"aaaa     70  65  67  60\nbbbb     55  60  55...
```

The `\n` character is the **newline** character

lines

The function **lines** gives us a list of lines

```
ghci> lines "line 1\nline 2\nline 3\n"
["line 1","line 2","line 3"]


ghci> file <- readFile "marks.csv"

ghci> lines file
["aaaa    70  65  67  60",
 "bbbb    55  60  55  65", ...
```

unlines

The **unlines** function does the opposite

```
ghci> unlines ["line 1", "line 2", "line 3"]
"line 1\nline 2\nline 3\n"
```

```
ghci> unlines . lines $ file
"aaaa     70 65 67 60\nbbbb     55 60 55 65
```

Parsing the file

Using **words** and **lines** we can parse the file

```
ghci > let parsed = map words . lines $ file

ghci > parsed
[["aaaa","70","65","67","60"],
 ["bbbb","55","60","55","65"],
 ["cccc","40","40","40","40"],
 ["dddd","80","60","75","60"],
 ["cccc","0","0","0","100"]]
```

Getting the averages

```
average :: [String] -> Float
average [student, a1, a2, a3, ct] =
          (read a1 + read a2 + read a3 + read ct) / 4


ghci> let averages = map average parsed
ghci> averages
[65.5, 58.75, 40.0, 68.75, 25.0]
```

Getting the student names

```
name :: [String] -> String
name [student, _, _, _, _] = student


ghci > let names = map name parsed
ghci > names
["aaaa","bbbb","cccc","dddd","cccc"]
```

Creating the report

```haskell
report_line :: String -> Float -> String
report_line student average =
        student ++ " " ++ show average
```

```
ghci> let zipped = zipWith report_line names averages
ghci> zipped
["aaaa 65.5",
 "bbbb 58.75",
 "cccc 40.0",
 "dddd 68.75",
 "cccc 25.0"]
```

Writing the output file

```
ghci > unlines zipped
"aaaa 65.5\nbbbb 58.75\ncccc 40.0\n..."


ghci > writeFile "report.csv" (unlines zipped)
```

All in one function

```haskell
report file =
    let
        parsed   = map words . lines $ file
        students = map name parsed
        averages = map average parsed
        zipped   = zipWith report_line students averages
    in
        unlines zipped
```