COMP105 Lecture 24

Useful IO Functions

Useful IO functions

print is the same as `(putStrLn . show)`

```
print_stuff = do
    print "hi"
    print 1
    print [1,2,3]
    print False
```

Useful IO functions

putStr prints a string **without** a new line

```
print_three a b c = do
    putStr a
    putStr b
    putStr c
    putStr "\n"
```

```
ghci> print_three "one" "two" "three"
onetwothree
```

Useful IO functions

readLn gets a line of input and then calls `read`

```
readLn' :: Read a => IO a
readLn' = do
    x <- getLine
    return (read x)
```

```
add_one :: IO ()
add_one = do
    x <- readLn
    putStrLn (show (x + 1))
```

Useful IO functions

forever repeats an IO action forever

- It's in the `Control.Monad` package

```
ghci> import Control.Monad

ghci> forever (putStrLn "hi")
hi
hi
hi
hi
...
```

Interactive code with `forever`

```
import Control.Monad
import Data.Char


process :: IO ()
process = do
    putStrLn "Give me some input: "
    l <- getLine
    putStrLn (map toUpper l)


main = forever process
```

sequence

`sequence` performs a list of IO actions

```
ghci> sequence [getLine, getLine, getLine]
one
two
three
["one","two","three"]
```

The final line is the return value of `sequence`

```
sequence :: [IO a] -> IO [a]
```

sequence

`sequence` works well with `map`

```
ghci> sequence (map print [1,2,3])
1
2
3
[(),(),()]
```

mapM

Alternatively, you can use `mapM`

```
mapM :: (a -> IO b) -> [a] -> IO [b]
```

```
ghci> mapM print [1,2,3,4]
1
2
3
4
[(),(),(),()]
```

when

`when` executes an IO action **if a condition** is true

```
ghci> when True (print "hi")
"hi"

ghci> when False (print "hi")
ghci>
```

```
when :: Bool -> IO () -> IO ()
```

unless

`unless` executes an IO action if a condition is **false**

```
ghci> unless True (print "hi")

ghci> unless False (print "hi")
"hi"
```

```
unless :: Bool -> IO () -> IO ()
```

Exercises

1. Use `readLn` and `print` to write an IO action `timesTwo` that asks the user to input an integer, and then prints out two times that integer

2. Use `forever` to write an IO action `ltFive` that repeatedly asks the user to input lines of text. If the line has fewer than five characters the program should output `yes`, and otherwise it should output `no`

3. Use `sequence` to write an IO action `addThree` that asks the user to input three numbers on different lines and then prints out the sum of those numbers