PROGRAMMING III (COMP 2209)
SEMESTER ONE, 2018

Home

Exercise Sheet Eight : Input/Output in Haskell

The aim of this tutorial is for you to become familiar with the use of IO actions in Haskell. This should include using basic actions as well as the "do" sequencing notation for actions.

Attempt the exercsise in your own time before Tuesday's lab and use that session for obtaining feedback and for asking questions related to the exercises.

Exercise One

The function `sequenceIO_ :: [IO a] -> IO ()` takes a list of actions, executes them all in sequence discarding their result values and finally returns the unit value. Provide an implementation for this function and use it to define the function `putStrLn :: String -> IO ()`

Exercise Two

Consider the following Haskell implementation of the N Queens problem. The function `queens` calculates all solutions and returns them as a `[[ Int ]]`. Modify the code as follows:

- Allow the user to input what size board to use rather than hardcoding the program to 8
- Pretty print the solution so that each solution appears as a grid something like the grid below for board size 6. I've used unicode character '\x265B' here.

```
---------------------
| . | ♛ | . | . | . |
---------------------
| . | . | . | ♛ | . |
---------------------
| ♛ | . | . | . | . |
---------------------
| . | . | ♛ | . | . |
---------------------
| . | . | . | . | ♛ |
---------------------
```

Exercise Three

Define an action `adder :: IO ()` that reads a given number of integers from the keyboard, one per line, and then displays their sum. For example:

```
> adder
How many numbers? 5
1
3
5
7
9
The total is 25
```

Do this firstly by definining keeping a running total as you read each number. It is helpful to define an auxilliary function to achieve this.

Exercise Four

Similar to Exercise One, the function `sequenceIO :: [ IO a ] -> IO [ a ]` takes a list of actions of the same type and executes each but this time it returns the resulting value from each action in a list of type `[a]`. Provide an implemenation for `sequenceIO` and then use it to provide an alternative implementation of `adder` in Exercise Three.

Exercise Five

Write an action `getLineDel :: IO String` that behaves as `getLine` except that it allows the user to delete a character that they have entered. The delete character is '\DEL' and the control character to move the cursor back one space in a terminal is '\b' (or '\BS') (nb. the backspace control character will show as the actual control character rather than moving the cursor in GHCi).

Exercise Six

Consider this implementation of the Hangman game presented in the lecture. Modify the code as follows:

- Fix a number of allowed guesses to be some function of the length of the input word. Display a "too many guesses" message and terminate the program if the second player exceeds the number of allowed guesses.
- Provide a dictionary file of secret words. Rather than having two players, have the program randomly choose a word from the dictionary as the secret word to guess.