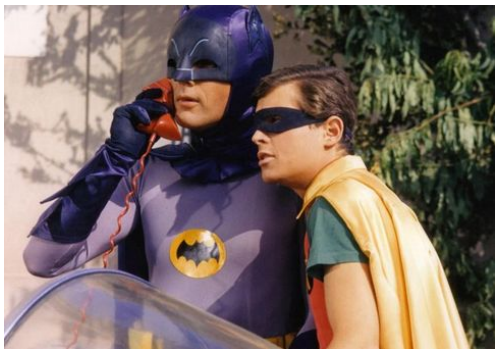
CS256 Coursework 2016-17 Part 2

Steve Luci Matthews

Date issued: November 10th 2016 (Thursday, Term 1, Week 6)
Last update: 10/11/16



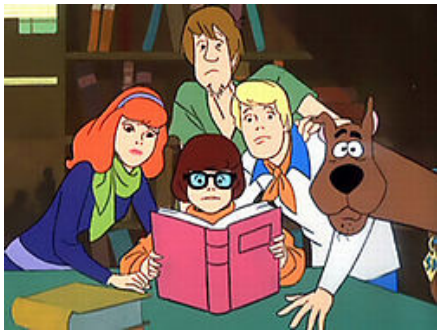
Robin! Its Scooby-Doo and Shaggy!
They want a parser generator for FP in FP!

This is the second of two parts which comprise the coursework for CS256. *Part 2* covers the following themes of FP:

1. Functional programming with higher order functions,
2. Analysing a single problem and designing a solution using FP,
3. Designing your own programming language.

In Part 1 of the of the coursework we focused upon *first order programming* in FP. Now it is time to move beyond the first order approach, and perceive the full power of FP that can be embraced using *higher order functions*. Creating your own programming language constructs is a good way to practice specifying and defining higher order functions. For the problem specified below prepare an answer in the form of a well commented, well structured, well typed Haskell program, placed together in a **single file** having the name `ass16-2.hs` , and submitted online using *Tabula*. Your submission will be individually visually inspected and assessed by the module organiser for the highest standards of functional programming practice, tested upon the Hugs implementation as installed in the DCS Linux systems (rooms CS001 and CS006), and individual feedback returned. Remember to help **me** the reader to help appreciate your work by providing excellent comments of the form,

```
{- Explanation on something
   ========================
   Explanation goes here ...
-}
Haskell code for something goes here ...
```

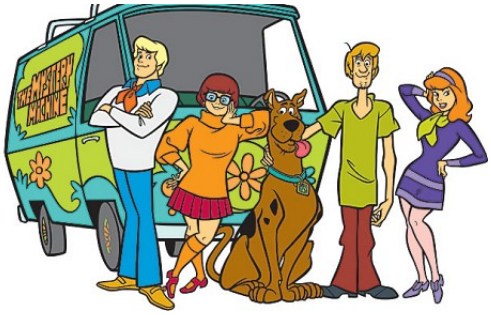


Daphne, Velma, Shaggy, Fred, and Scooby-Doo

The coursework

The coursework is to design and implement a parse tree generator for a small functional programming language called *Shaggy*. The grammatical constructs of Shaggy are:

1. A Shaggy **constant** is an unsigned decimal numeral such as `42` or `063`
2. A Shaggy **variable** is a finite alphanumeric string beginning with a letter. All Shaggy variables are assumed to be of the same integer type. Each variable (one at a time) is declared by the syntactic form `var` : *var*
3. A Shaggy **program** is a finite sequence of Shaggy **definitions**
4. A Shaggy **definition** has the form *var* = *exp* where *var* and *exp* are a Shaggy variable and Shaggy expression respectively
5. A Shaggy **minus** expression has the form - *constant* or - *exp*
6. For each Shaggy expression *exp* , (*exp*) is a **bracketed** Shaggy expression
7. A Shaggy **addition** is of the form *var* = + *exp exp*
8. A Shaggy **multiplication** is of the form *var* = * *exp exp*
9. A Shaggy **evaluator** is of the form `eval` *exp* where *exp* is either a Shaggy constant or variable. Note that the task of this coursework is to design and implement a parser generator for Shaggy, and **not** to build an evaluator.



Fred, Velma, Scooby-Doo, Shaggy, and Daphne

Some examples of Shaggy programs are:

- ```
  var x
  x = - 42
  var y
  y = + x 1
  eval y
  ```

- ```
  var x
  x = 4
  var y
  y = - 3 x
  x = + (- y 1) 2
  eval x
  ```

Additional notes

- Your work may assume that each Shaggy variable is defined at most once in a Shaggy program.
- All Shaggy programs are assumed to be syntactically and gramatically correct.

Assessment Criteria

Your work will be examined for its good use and application of functional programming principles and their expression in Haskell. Your final mark for this coursework (of 100%) will be broken down as follows. 30% for your correct **analysis** of the problem. 20% for your appropriate **algorithm** used to implement the problem. 30% for your choice of **language** constructs (coded in Haskell) used to represent the algorithm. 20% for your demonstrative **testing** of key language constructs in your program.

Submission procedure

Submit your solution to this assignment on-line as a single Haskell source file named `ass16-2.hs` on Tabula. Your submission must be made by **12:00 noon** of **Thursday December 8th 2016 (Term 1, week 10)**. This assignment counts for **25%** of your total credit for the module CS256. In accordance with University guidelines, 5% per day (or part thereof) of the maximum possible mark for this assignment will be subtracted from your mark for a late submission.

Plagiarism Detection

Each student is politely reminded that the CS256 coursework is an individual assignment whose contents is presumed, unless otherwise acknowledged, to be only the work of that person. Each submission received will be automatically compared with each and every other such submission using plagiarism detection software. The University's disciplinary procedures **will** be applied to each student whose submission is suspected, within all reasonable doubt, to be the unacknowledged result of either copying from or collusion with another person or persons within or without of Warwick University.