B5 - Advanced Functional Programming

B-FUN-500

Functional EvalExpr

A better way

2.22

Functional EvalExpr

binary name:	funEvalExpr
repository name:	fun_evalexpr
repository rights:	ramassage-tek
language:	Haskell
compilation:	via Makefile, including re, clean and fclean rules
build tool:	stack wrapped in a Makefile *(see below)*

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

You should already know what an EvalExpr is.
So implement a functional one, that is able to parse a string given as argument and display the resulting value, followed by a new line.

Terminal

```
~/B-FUN-500> ./funEvalExpr "3 + 5.34"
8.34
~/B-FUN-500> ./funEvalExpr "(0.345+ 5 )*( -2-1) / 3"
-5.35
```

The output must always have two digits after the decimal points.
For instance `1.00`, `3.14`, `4.50`, `0.67`.

Your program must handle **float numbers**, **parentheses** and the following binary operators:

1. sum (+) and difference (-)
2. product (*) and division (\)
3. power (^)

The list above is sorted by precedence, from lower to higher.

You may implement a recursive descent parser with combinators, following a Parsing Expression Grammar (PEG). To go further, a packrat parser is a nice thing to have.

Coding Style

There is now a coding style for Haskell projects which you must respect. The document for this coding style is found alongside this project.

Stack

Stack is a convenient build tool/package manager for Haskell.
Its use is required for this project, with **version 2.1.3 at least**.

It wraps a build tool, either **Cabal** or **hpack**.
You are required to use the hpack variant (package.yaml file in your project, autogenerated .cabal file).

This is what stack generates by default with `stack new`.

Stack is based on a package repository, **stackage**, that provides consistent snapshots of packages.
The version you use must be in the **LTS 16** series (`resolver: 'lts-16.15'` in stack.yaml).

In `stack.yaml`, extra-dependencies cannot be used.

base is the only dependency allowed in the `lib` and `executable` sections of your project (package.yaml).
There is no restriction on the dependencies of the `tests` sections.

You must provide a **Makefile** that builds your stack project (i.e. it should at some point call 'stack build').

'stack build' puts your executable in a directory that is **system-dependent**, which you may want to copy.
A useful command to learn this path in a system-**independent** way is:
`stack path --local-install-root`.