HASKELL CSV ASSIGNMENT WILL BROWNE

Student number: 09389822

To get my program up and running, just compile and type "main" in the prompt!

The list of functions that my program supports:

- **insert**
- **select**
- **delete**
- **reformat**
- **datefix**
- **gridfix**
- **distinct**
- **count**
- **list**
- **update**
- **report**
- **help**
- **save**
- **? (help)**
- **quit**
- **q (quit)**
- **show,**
- **print**

The details of these functions can be found by typing *help* in the program!

Representation of the spreadsheet

I decided I would represent each value in the CSV singly.

```
data DataVal = DataVal {
    x :: Int,
    y :: Int,
    col :: [Char],
    val :: [Char]
  } deriving (Eq)
```

Where x is the column, y is the row, col was a value I was going to use to store the column name but never eventually used, and val is the actually value of the cell.

Note: For my CSV - X starts from 1 and Y from 0

Then came my main program, which is passed around to functions.

```
data Program = Program {

            csv :: [DataVal],

            cur :: [DataVal],

            state :: Bool,

            file :: String

   } deriving (Eq)
```

CSV is the parsed CSV file (a list of DataVals), cur is the currently selected rows (a list of DataVals), state is whether the parsed CSV file is saved, and file is the name of the filename.

All my pure functions have the same type signature which allow them to be listed as dispatch functions and all be called in the same way from the mainline program.

```
dispatch :: [([Char], [Char] -> Program -> IO (Program, [Char]))]
dispatch =  [ ("insert", liftUp insert'),
            ("select", liftUp select),
            ("delete", liftUp delete'),
            ("reformat", liftUp reformat),
            ("datefix", liftUp (datefix monthsList)),
            ("gridfix", liftUp gridfix),
            ("distinct", liftUp distinct),
            ("count", liftUp count),
            ("list", liftUp list),
            ("update", liftUp (update columns)),
            ("report", report),
            ("help", liftUp help),
            ("save", save),
            ("?", liftUp help),
            ("q", quit),
            ("quit", quit),
            ("show", liftUp show'),
            ("print", liftUp print')
           ]
```

liftUp is a function that takes will promote existing functions so that they have their correct type. This is based on the function *lift* in the Control.Monad library.Some of the functions have very slightly different type signatures because they take one more argument. This was easily resolved through the use of partial functions.

Example ("update", liftUp (update columns)), Where ‘columns’ is the column list in my program that I need to pass to update to make sure that a user can insert into a column by stating a column name.

Note – because I ended up doing the update function near the very end of the project, I only introduced the idea of being able to reference a column by its name and also just number in this function. Realistically it should be available in some other functions, but I just didn’t get around to doing it.

Also the same goes for regular expressions. This only works in the select function and there’s no reason why it shouldn’t work in others. However these other functions were already functioning and happy as they were, so I didn’t change them.

These are changes that are easily done since my regular expression functions are in stand-alone functions, and column list can be simply passed to these functions as an extra argument and changed in the dispatch command list.

Error handling

For bad user input I have all user input being caught with if statements in case they aren’t sensible, are null etc.

My exception handling is used in the load function in the form of a catch. This will prevent the program from crashing out if something went wrong whilst trying to load a file.

Sort

Sort is the only function I didn’t managed to implement. My progress was made difficult because of the way my CSV (once parsed) was represented (just a straight up list of cells rather than by list of rows or columns). I figured the best way to tackle the sort would be to have my CSV represented as a list of rows, as it would be much easier to swap rows in this way. My attempts at this however made me completely lose faith as my function looked like it would be terribly slow and just straight up bad programming. I have the bones of simple sort in place which uses the built in sort function to sort a single list of DataVals. However if I was to tackle the nested sort in the specification, I would use quick sort.

Note: There may be an encoding error when trying to compile

```
lexical error in string/character literal (UTF-8 decoding error)
```

I developed the majority of the project on my Mac but when in college on the windows machines, I think the format may have gotten slightly messed up. I get this while in Notepad++ but only need to change the encoding to UTF-8 to get it up and running!