Assignment 2: Syntax aware Editor

The syntax aware editor is a ncurses application, that formats the text in the console window. For that the terminal must support at least 256 colors. This assigment is programmed in **Haskell**.

Prerequisites

- Stack is used to build the haskell application.
- Besides the haskell dependencies, which are managed by stack, also `ncurses` is needed and must be installed
- For some linux distributions (like Arch) some symlinks of the ncurses header files have to be created since the haskell dependency requires `ncursesw` header files

Limitations

- No tabs are allowed in the documents
- Lines must be smaller then the editor width
- Resizing the terminal window smaller then 40 columns may crash the editor

Usage

- The cursor can be navigated directly with the arrow keys
- `F2` Is used to save the contents of the editor into a file (Saving only works if the file was opened at startup)
- `F12` Quits the editor without any check if contents of the editor changed since the last save
- There is optional mouse support that can place the curosr at the clicked position

How to build / Start the editor

To build and simply run the editor `stack` is used:

```
stack run <filepath>
```

Note: If the filepath is not given a empty document is opend, but can not be saved to disk.

Run without stack

There is a static build version of the editor available, this version should run on Ubuntu 18.04 LTS and newer systems. The terminal must support at least 256 colors otherwise this binary does not work. (Doesn't work in WSL)

```
./editor-static <filepath>
```

Architecture

Since the NCurses library is used the application logic runs in a event-dispatch loop that will parse the text into a AST structure, which then is rendered to the NCurses window. After that the loop will wait for some events to occur like key or mouse events that will modify the overall state of the editor.

Because transforming text into an AST is a very complex task the parsec library is used to parse the text into the respective tokens, which are then used to build the AST. The tree does not only contain the tokens, but also the whitespaces around them so that the text can be rendered back to the window without destroying the original text structure.

For rendering the text to the window with the correct highlight the `EditorASTState` object is used to keep track of a virtual cursor position. This is necessary since each node of the AST can be seen as a list of draw commands and they do not hold any absolute position, which can be used to decide if the tokens are in view or not. The draw function of the `CursesFormatter` can therefore be seen as a recursive monadic fold operation. This operation is performed twice since in the first run the correct identifier, which should be highlighted, is enumerated and the second run is used to actually render the contents.

Test protocol

Testing the syntax highlighting is a very difficult task, since the curses environment has to be mocked completely. Therefore we decided to only use integration tests for the editor with a fixed set of input files and verify that the syntax highlighting is according to our specifications.

#	Filename	Test result
1	complex_function.fprog	OK
2	complex_function_without_parantheses.fprog	OK
3	complex_records.fprog	OK
4	complex_records_nested.fprog	OK
5	simple_expression.fprog	OK
6	simple_expression_cond.fprog	OK
7	simple_function.fprog	OK
8	simple_records.fprog	OK

Besides, these test cases also the basic editor functionally has been tested. The following table does list these test cases:

#	Test case	Test result
9	Use Arrow Keys to navigate the cursor	OK

#	Test case	Test result
10	Input some characters into the editor	OK
11	Use F2 to save opend text file	OK
12	Use F12 to exit the editor	OK
13	Use mouse to position cursor in text	OK (1)
14	Process text longer then editor	OK
15	Process text wider then editor	Faulty (2)
16	Test braces matching	OK
17	Test name highlighting	OK (3)

- **(1)**: Requires system support
- **(2)**: Ncurses auto inserts line breaks that break formatting
- **(3)**: Global highlighting only