CS 352 HW 5: Error Handling

Due Wednesday, April 8, 10pm

This homework is a rework (in two different languages) of part of the Haskell homework with better software engineering principles. The input format is the same as HW 2 (repeated below), but this time you may not assume that the input is correctly formatted. The only part of HW 2 you are writing is `computeGpas`, but this time you are required to correctly handle input format errors.

Input Format

The input consists of a sequence of lines, where each line consists of seven space-separated words that give the grade for a student in a particular course. For example, the line below indicates:

```
PHL 220 A 3 Wai Choi B
```

- *Subject* (e.g. PHL): one or more uppercase letters
- *Course number* (e.g. 220): a non-negative integer
- *Section* (e.g. A): a single uppercase letter
- *Number of credits* (e.g. 3): a non-negative integer
- *First name* (e.g. Wai) and *Last name*: each a sequence of letters and - (hyphen) or ' (apostrophe) characters; you may assume that no student has a first name or last name longer than 15 characters.
    - Should be combined together in `ClassRecord` as "`Last-name, First-name`"
- *Grade* (e.g. B): Must be a valid grade; A/B/C/D/F, where A-D have an optional trailing + (plus) or - (minus). A is 4.0, B is 3.0, C is 2.0, D is 1.0, F is 0.0, + or - add or subtract 0.3, respectively.

Possible Errors

There are a variety of possible format errors in this specification, but your code will only be required to handle four possible errors:

- `InvalidCourseNum` or `InvalidNumCredits`: The course number or number of credits, respectively, is not an integer.
- `InvalidGrade`: The grade does not match the format above.
- `WrongFieldCount`: There are not exactly seven whitespace-separated fields in the input line.
    - In cases where `WrongFieldCount` or another error could both apply, your code is allowed to return either, as you like, e.g. `InvalidCourseNum` or `WrongFieldCount` for either of the following lines:

```
PHL wrong
PHL wrong A 3 Wai Choi B extra stuff
```

Running Code

In this assignment, the provided `run_hw5` source file for each language handles I/O, line reading, and printing, with the same command line arguments as HW 2:

- If there are no command-line arguments, it reads from standard input and writes to standard output (Ctrl-D on Linux or Ctrl-Z on Windows will end standard input).
- If there is one command line argument, it reads from that file, and writes to standard output.
- If there are two command line arguments, it reads from the first file and writes to the second.

Each can be compiled using the usual compilation commands for its language:

```
g++ -std=c++14 -o hw5_cpp run_hw5.cpp
ghc -o hw5_hs run_hw5.hs
rustc -o hw5_rs run_hw5.rs
```

Question 1

For C++ and <u>one</u> of Haskell or Rust, complete the functions `letterToGpa()` and `parseRecord()` in the starter code (the Rust functions are named in `snake_case` to satisfy the compiler-checked style guide).

[6 (each language)**]** `letterToGpa()` converts the grade string to a floating-point GPA according to the rules above. If it fails, it throws an `InvalidGrade` exception in C++, while the Rust and Haskell versions use a variant wrapper type with the following names:

Language	*Type*	*Success*	*Failure*
Haskell	`Maybe Double`	`Just` *`value`*	`Nothing`
Rust	`Option<f32>`	`Some(`*`value`*`)`	`None`

[15 (each language)**]** `parseRecord()` parses the entire line in the input string, producing a `ClassRecord` in each language. If the C++ version fails, it throws an appropriate `ParseError` exception, while the Rust and Haskell versions use variant types with the following names:

Language	*Type*	*Success*	*Failure*
Haskell	`Either ParseError ClassRecord`	`Right` *`record`*	`Left` *`error`*
Rust	`Result<ClassRecord,ParseError>`	`Ok(`*`record`*`)`	`Err(`*`error`*`)`

Style

[6] Your code should not be significantly more complex than necessary, and should be structured in a way that facilitates understanding. Make use of whitespace and comments as appropriate to make your code clearly readable. Function and variable names should clearly represent their purpose or be sufficiently limited in scope to be clear from context.

Question 2

[6] Choose one of the details of the input specification not covered by the four possible errors in the assignment, and modify your code and the starter code for both languages to check that formatting detail as well.

- Add an appropriate `ParseError` type for the new error.
- Modify the handling code in `run_hw5` to handle it.
- Leave a comment at the top of each file describing the formatting error you checked for.

Submission Instructions

Your code for Questions 1 & 2 should be put into a zip file; include all the starter code for the two languages you used that was provided in `cs352_hw5_starter.zip`.

The zip files should be submitted on Moodle by the assignment due date.