CS3820: Midterm Project

Due October 15th by midnight

Introduction

There are two parts to this midterm project, both equally weighted. You will have two weeks to complete these parts, both of which are due October 15th at midnight. Late submissions will not be accepted, even if it is late by a few minutes. Start early and make sure you submit before the dropbox closes.

Part 1: programming portion

Implement the undefined functions (similar to the homework assignments) in the file `parse.hs`. You will be given an autograder to help you verify your progress. As a reminder, you will receive a 0 on your submission if we cannot compile it!

The following types are defined in `Autograder.hs` and are the types you should use in your solutions.

```
-- All possible tokens of an expression string
data Token = LPar | RPar | Literal String | Plus | Minus | Asterisk | Slash deriving (Eq, Show)

-- Denotes the operation being performed in a binary expression
data Op = Add | Sub | Mul | Div deriving (Eq, Show)

-- Represents an expression as either a binary operation or a double value
data Expr = Binary Op Expr Expr | Val Double deriving (Eq, Show)

-- Represents the possible errors encountered while running an expression.
data RunError = MismatchedParentheses
              | LexFailure LexError
              | ParseFailure ParseError
              deriving (Eq, Show)

-- Represents an error encountered during lexing
newtype LexError = UnknownToken String deriving (Eq, Show)

-- Represents an error encountered during parsing
newtype ParseError = SyntaxError String deriving (Eq, Show)
```

- `validParens` – this function checks that the given expression has valid parentheses. Valid in this context means that the parentheses have matching closing braces. For example, `(()())` is valid while `(()` is not. Note that the input string will contain more than just parentheses.
- `lex` – this function splits your input expression into tokens. It should return either a `LexError` or a list of tokens (`[Token]`). **Hint**: you may find it helpful to make sure '(' and ')' are surrounded by spaces prior to lexing.
- `parse` – this function takes your list of tokens and converts them to an expression (possibly a compound expression) or returns a `ParseError`. **Hint**: I think the simplest approach for the midterm is to reverse to token list and parse from the end. This is not a correct way to parse in general, but works for this setting. In the coming weeks we will learn about recursive descent parsing, which is both more robust and correct.

- `eval` – evaluates the given expression. This function does not return an error.
- `run` – this function takes an input string that represents an expression and returns an output string containing either the result of running the program or an error message. You can think of this function as our "expression interpretter."

Part 2: writing portion

Summarize the paper "Fundamental Concepts in Programming Languages" by Christopher Strachey. The goal is to summarize the paper clearly and concisely. You will be graded on the quality of your writing and how well it demonstrates your understanding of the paper, *not* on length. Please don't try to do this without reading through the paper a few times. We will only accept a PDF for your report.

To help you think through this task, here are a few things to consider:

- What are the key concepts the author is trying to communicate?
- What exactly are these concepts? Can you explain them clearly and concisely? Note "clearly and concisely" does not mean reiterate the the definition.
- Can you relate some of the conent of this paper to a programming language you know?
- How do these concepts all fit together?

Your goal here is to demonstrate that you have read the paper, understood what you have read (at least most of it), thought about what you've read, and are able to communicate what you have read.

I imagine many of you will be new to reading this kind of paper. So here are a few reading tips:

- Don't expect to understand everything in one read. You will likely need to read the paper several times before everything clicks.
- Don't spend too much time on things you don't understand during your initial read and especially don't spend time on understanding formulas or equations. Spend more time on the challenging parts on the following read throughs.
- Feel free to ask us questions clarifying your understanding of the paper or certain concepts.