CS381 Project Milestone

Teammate:

Mingming Su

Zhidong Zhang

Chia-Yu Tang

Yihong Liu

1. **Introduction**
    - **What is the name of your language?**

      Calculator Haskell Program

    - **What is the language's paradigm?**

      Imperative language view computation

2. **Design**
    - **What *features* does your language include? Be clear about how you satisfied the constraints of the feature menu.**
        - **i. Basic data types and operations**

          Data bool = True | False

        - ii. **Conditionals**

          If….then…. Else error (a function which exclude input error)

        - iii. **Recursion/loops**: **expressionWithParensOrNumber** is a function that we will use in chainMultiplyDivide function.

          chainMultiplyDivide :: Parser Float

          chainMultiplyDivide =

              chainl1 **expressionWithParensOrNumber** (divide <|> multiply)

        - iv. **Variables/local names**: We use where statement to define local variables.

plus = fmap (const add) plusChar

  where add a b = a + b

        plusChar = char '+'

v. **Procedures/functions with arguments**

   expression :: Parser Float

   expression = chainl1 chainMultiplyDivide (plus <|> subtract)

vi. **User-defined data types and pattern matching(3)**: We will have fmap to map over the char '+' with an operator.

   plus :: Parser (Float -> Float -> Float)

   plus = **fmap** (const add) plusChar

     where add a b = a + b

           plusChar = char '+'

- **What *level* does each feature fall under (core, sugar, or library), and how did you determine this?**
  i. **Basic data types and operations**

     Core, It's the safety properties of the program. It belongs to core features.

  ii. **Conditionals**

     Syntactic sugar, we can use different features to replace it.

  iii. **Recursion/loops**

     Syntactic sugar. If the function was removed, it could be rewritten in another way using different features.

  iv. **Variables/local names**

     Syntactic sugar. If the function was removed, it could be rewritten in another way using different features.

  v. **Procedures/functions with arguments**

Library-level, It is not really a language feature at all. ‘chainl1’ depends on the haskell library.

vi. **User-defined data types and pattern matching**

Syntactic sugar. ‘Fmap’ helps us transfer char ‘+’ into an add operator.

- **What are the *safety properties* of your language? If you implemented a static type system, describe it here. Otherwise, describe what kinds of errors can occur in your language and how you handle them.**

  If the user input some string without definition in our program, such as 2! and 2*x , it will cause error, we will write a pattern matching to judge if the input of string or float is valid to compile. If not, print an error message to request re-enter.

3. **Implementation**
   - **What *semantic domains* did you choose for your language? How did you decide on these?**

     float-> float-> Maybe float

     We will need the user to input at least 2 numbers and one operator to do the calculation. Numbers can be any float. The result can be float or error. Error occurs when the user’s input is invalid.

   - **Are there any unique/interesting aspects of your implementation you’d like to describe?**

     We use the “Parsec” to write the code which is really useful to make the code more clear. By using Parsec, we could combine some small, sample parser to complex parser. For example, we have a parser “+” and a parser “number”, and after combing them, we get a add function parser.