# Prefix Calculator Write-up

## Approach:

My approach to this project was to first map out pseudocode for how I would write the program in an OOP language. Then, I could ‘translate’ it to Haskell code, which I found was much more difficult than anticipated. This was mainly due to inexperience with syntax, structuring, and built-in functions provided by Haskell. Because of prior experience in evaluating postfix expressions in Java, I had a good idea of the general program flow, I just needed to figure out how to execute that in Haskell.

## Organization:

The program is made up of 3 parts: 1. User input/validation, 2. Expression evaluation, 3. Main loop and history handling. Completing the program each step at a time set a solid foundation for the next step and made making changes to previous steps much easier to figure out. I will talk about this more in problems I encountered. Since each part is connected in some way, going step by step seemed much simpler than trying to program every functionality at once and making adjustments on the fly. Writing out pseudocode for the main functions also helped a lot to understand the program flow and how everything was going to connect. While I ended up needing to write functions that I did not plan to from the start, they were mostly small, helper functions.

## Problems:

A big problem I encountered was figuring out how to implement and integrate the history list. When I started setting up the program structure, I did not think about how implementing history would work. Originally, my program allowed the user to enter an expression and it would exit after. However, this would not allow the program to save a list of history values as it would always be reset every time the program was executed. To fix this, I made the program loop until the user entered a specific input to close it so that the history values would be saved and could be accessed. For integrating history, while this was not as difficult due to having all the other parts of the program working, it was still a challenge to see which functions needed to access the list and how to manipulate it. For example, I didn’t realize my main loop needed the history list and only the evaluation functions would need it, but then the list wouldn’t update after each expression. Another general problem I had was figuring out how to utilize different built-in functions, such as using the ‘just’, ‘case’, etc. structures due to unfamiliarity with things such as syntax and scope of the functions. This problem was solved with lots of trial and error and searching online for examples and figuring out how to utilize them properly.

## What I Learned

The main thing I learned through this project was how to manage a relatively bigger program while working with a functional language. Compared to previous homework assignments, this project took a lot more planning out and trial and error. My previous experience with a similar problem, just in a traditional OOP language helped speed the process of understanding the project flow. However, implementing it in Haskell was a challenge and was where I learned the most. While the language still feels weird to use, I am definitely more comfortable with the syntax than before after having experienced using different structures such as ‘just – maybe’, ‘let-in’, and ‘case’ and just overall getting to use it more.

## What I Was Unable to Implement

I did not use the !? operator to check the history id values, I am not sure if that was a requirement. When I tried to implement it, there were many import options, some causing more errors in my code so I chose not to use it.