Write up Project 1

- Project approach
    - For this project my primary object was to make a calculator that will have support the usage of prefix notation in Haskell also with the use of added history functionality. After doing a thorough analysis of the project, I noticed the components for example parsing, tokenizing, history management. Haskell has a strong type of system and functional paradigms that helped create a robust maintainable solution
- Project organization
    - This project was implemented in one single file, which is called PrefixCalculator.hs, with the scope of the project, this proper organization made it clear that it allowed for the easy modularization, this the key sections of this file will include
        - Defining data by using expr data type to show different types of expressions
        - The main function handles the loop and the UI for the calculator
        - Tokenization, where the section breaks down the specific input strings into things called tokens such as operators, numbers and history references
        - Helper functions are used to handle tasks such as input trimming and error management
- Problems encountered
    - Handling partial functions
        - Using head and tail functions posed a risk of runtime errors when lists were empty which will generate compiler warnings
    - History management
        - Where to ensure the correct mapping which is between the history ids and this resulted into a challenge, with history list is stored in reverse order
- Solutions fixed
    - Refracting head and tail function with pattern matching to safely handle list operations to avoid runtime errors and thus eliminate complier warnings
    - The history was stored with the most recent result first and I reversed the list when accessing, this ensured correct history id mapping
- Lessons learned
    - This project help me reinforce the important understanding of immutability, recursion and thus having strong typing in a functional programming language. To avoid partial functions, improve the safety and when designing recursive algorithms for future parsing and evaluation to enhance my strong understanding of functional problem solving.