- How did you approach the project?
    - I approached this project by first coding it in Java, the language I was most familiar with. Doing this allowed me to understand all the functionalities of this calculator, how prefix notation works, what the project is asking for, and overall clarity on the project. I next attempted to reconstruct this code in Haskell but quickly realized that I was stuck in an imperative paradigm. I spent a few hours thoroughly understanding Haskell and how to essentially unlearn the paradigm I was currently stuck in to relearn a new skill set. As I was learning, I was able to slowly evolve my code into parts that worked individually. Eventually, I was able to combine the multiple parts of my project to produce one full-functioning project.
- – How did you organize the project? Why?
    - I organized the project into the following sections:
        - expression data
        - parsing through
        - eval expression
        - printing
        - updating hist
        - evaluation loop
    - The reason I organized the project into these sections was to divide the project into multiple subtasks and complete each section at a time. This allowed for easier testing and less severe problems. There was a point where I had to restart the project because I attempted to solve all of the sections at once and I was unable to connect all the subtasks together. Thus, these sections allowed me to complete this project most effectively.y.
- – What problems did you encounter?
    - The biggest problem I encountered was getting around not using a specific data structure, like a stack, and not using mutable variables. It was difficult to wrap my head around how to solve this problem another way. Most of my errors were compile-time errors, and it had to do with types not matching. I had little to no runtime errors - the only ones seemed to do with saving and properly returning history.
- – How did you fix them?
    - I was able to fix these problems by simply learning more about Haskell. Engulfing myself more and more into the functional paradigm and thoroughly understanding the functions that Haskell already has and the syntax that I can use helped me greatly to tackle my feeling stuck. Once I was able to understand how to create specific and all-encompassing functions, it was a lot more smooth sailing from there. To solve my runtime errors, I had to gain a better understanding of how lists worked in Haskell and how I could use the cons operator to solve this.
- – What did you learn doing this project?
    - I now have a far more complete understanding of functional programming and its benefits. I understand how programming functionally, although it can be more tedious, drastically helps reduce the potential for bugs.
- – If you did not complete some features of the project, why not?

    - I was unable to handle the divide by 0 feature, as I had only discovered this bug far too late into implementation and could not figure out a way to incorporate it within the code I had already. Since there seemed to be significant dependencies, it seemed difficult to include this restriction without any errors arising. Perhaps if I spent more time planning this project before I began coding, this mishap could have been avoided. In the future, I will thoroughly plan out a project before I start it. With my now better knowledge of Haskell, I will be able to map out all the use cases of a project, including all edge cases which always seem to bite me in the back.