CMPT 383 programming 4 report

#301448555

Chien An Chen (Edward)

Design choice:

I decided to rename some given variable names to have a better understanding of the function without modifying the actual function type. I also use some auxiliary functions to help with the general readability of the code and to implement algorithms with greater ease. Outside of them, I mostly follow the algorithm posted in the lecture slide to ensure correct results are achieved as much as possible. In this program, I used a lot of “case .. of ..” instead of the if then else statement since the former is easier to use in my opinion. There are a lot of brackets used in my program. Though I could have used the concept of currying through dot operator (“.”), I decided to not incorporate into my code due to my lack of experience in Haskell.

Features:

This program will take in a series of function expressions and performs type inference, then validates the function through the process. If the function expression is not an legal expression, the program will return “Type Error” as result, otherwise it will show the corresponding return type of the function expression. To test certain test cases, I implemented a test function in my programming for easier testing/debugging

Issue:

For me, this is the hardest assignment in the entire course span of CMPT 383. The amount of time I spent on this assignment is most likely the amount I spent on previous assignments combined. I struggle with using state monad and only figure out the correct function for getFreshTVar through trial and error. Even following the slide, the results produced by my program does not 100% match with the provided test cases and results. In some test examples, comparing to some other’s test result, I figured out that my TVar is not being properly substituted. It happened mostly in App or LetIn.