Daniel Elmar - 2209 Course Work Report 2022

Student ID: 31820638

Development Strategy

Implementation

When implementing the challenges I first read the provided Instruction' pdf section on the challenge multiple times thoroughly to ensure I had fully understood the expected behaviour, types of input, and any other information provided. I then did some background reading on the topic, this included; the provided links, wiki, google, and various forums. I also looked at any provided functions, for example, those in Parsing.hs. If possible I worked through the provided inputs and expected outputs for the challenge provided in the Instructions pdf. I slowly stepped through the input until I reached the expected output, this helped to understand the inner workings of a potential solution before I start coding, I used sketches, diagrams, and pen/paper to do this. This reduced the need to rewrite code. In some cases this was infeasible, such in the case of challenge 2. In some cases I also used tools to help me understand the challenge, for example for challenge 6 I used a Lamda Calculus Calculator to see how different expressions reduce.

After fully understanding the challenge I started to write high-level pseudo-code which describes what needed to happen for my implementation to work and what helper functions I need. I included special cases and necessary checks I would need. I kept my function names and variable names quite descriptive, which lead to some long names although I feel they aid in understanding and the readability of my code. I considered if I could write generic functions which accept a type ‘a’ and therefore increase the reusability of my code, I also considered any types synonyms I could create to make my code cleaner.

I then started to slowly replace my pseudo code with actual code, compiling my code to test after each helper function was finished and commenting my code where necessary.

Testing

I tested each helper function in isolation from the other functions, providing the necessary inputs directly. I tested different inputs to test each pattern match within the function I was testing along with anomalous inputs. This was very effective as it helped find and eliminate issues before they caused errors in other functions, which would have been harder to debug

In my Test.hs file I used IO() to run my tests, I used getCPUTime also to identify if any tests were taking too long to complete. I tested a variety of inputs including base cases, complex cases, and error cases, this was an effective method of testing.

Bibliography

Rathke, J., 2022. *38 Implementing Beta Reduction*.