Student ID: 30012627

6/01/2020

Programming 3: Coursework Report

The approach taken to solving the challenges in this coursework is as follows: Firstly, I read each challenge specification thoroughly and annotated it to understand the requirements. Then, I went through the examples in the tables one by one, i.e. how the right-hand side of the table is derived from the left-hand side. Following these initial steps, I developed a rough plan on paper, of how I would code the algorithm to solve the challenge. This plan included the basic overview of the functions I would need and their types. The next step was to break the main functions into helper functions that would each specialise into doing one specific task. Finally, I coded and developed my written solution and handled compilation and test case errors one at a time.

After making sure that the algorithm works for the given test cases, I created simple expressions that were not already covered by the given test cases, followed by more complex expressions for more complicated cases. I developed the test expressions keeping in mind the different data constructors of type LamExpr or LetExpr. For all my test cases in each challenge, I worked out the solutions on paper, before testing for the specific answers. I also made sure to check for boundary cases. For example in challenge 2, when the maximum number of reductions is reached, for any number greater than this the algorithm should give the same answer etc. The tests were added to the already provided test cases within the simpleTests function, in the Main program. They were then run using the 'main' command in the command line. In cases that I failed a test, I split the failed test into multiple parts and separately tested for these parts. This made it a bit easier to identify the source of some bugs.

In order to keep track of the changes I made and keep my work safe, I regularly committed my work to my GitHub repository. This also allowed me to easily retrieve earlier versions of the file whenever I wanted to do so. Given more time to develop my testing techniques, I would utilise H-Unit testing as it allows for specifying tests more succinctly, as well as labelling and grouping of tests (Hackage, 2020). These features would make it easier to create, change, and execute tests.

Word count: 386

References

Hackage. (2020). HUnit: A unit testing framework for Haskell. [online] Available at: https://hackage.haskell.org/package/HUnit [Accessed 6 Jan. 2020].