School of Electronics and Computer Science
Faculty of Physical Sciences and Engineering
University of Southampton

COMP2209
Coursework Challenges - Report

Version 1

Dzhani S Daud
StudentID: 30702321
dsd1u19@soton.ac.uk

11^{th} January 2021

1 Solution Implementation

For starters, I set up a quick private git repository to help me keep up with the changes in my code. Before attempting each challenge, I always spend a lot of time reading its instructions. That helped me get a better understanding of the problem, its requirements, and its context. I then tried different ways of breaking this problem into smaller issues. Now that I had divided everything into distinct parts, I started thinking of ways of implementing the solution. At some point, I had trouble writing the code in a 'functional programming way', because most of my experience was with imperative programming. However, as I delve deeper into Haskell, this problem eventually disappeared. After implementing each challenge I started refactoring the code to make it more readable and "pretty". I even made a bit of research on some of the best code-style practises. I also realised that a lot of the helper functions I was using were already defined in the base Prelude distribution of Haskell. So I was able to safely delete some parts of my code.

2 Testing

I started testing each challenge right after its implementation. I initially only used the "ghci" interface, but I soon realised this method is extremely inefficient. I installed HUnit, because it contained everything I needed to test my solutions. Using its already defined set of functions makes everything much more easy and efficient. HUnit also executes its tests independently of each other so if one of them raises an error, the next ones continue executing. After completing a challenge I always spend time thinking about its possible base and edge cases. I then wrote down the automated tests using the assert methods that HUnit offers. What was most great about automating the tests was that I was now free to edit the code without worrying that I might make a mistake and break something. This was particularly helpful when I was refactoring my code. There were also a few cases where the automated tests spotted a mistake in my solution.

3 Bibliography

Dean Herington (2002) HUnit Library (Version 1.6.1.0) https://hackage.haskell.org/package/HUnit