Assignment Part 8 – Prolog

# Program Testing Procedure

Execute fizzbuzz(1000) and observe output. Every 15th line should be “FizzBuzz”, every 3rd line should be “Fizz”, every 5th line should be “Buzz”, and every other line should contain the line number.

# Weekly Question

Fizz buzz was more difficult to implement in Prolog than the other languages. Fizz buzz is naturally conceptualised in a procedural manner: **loop** over numbers, **if** divisible by 3/5/15, **print**, etc. However, in Prolog, logic is the main functionality provided by the language, and the presence of any procedural/imperative elements is almost coincidental. Enacting a specific control flow (as is required for fizz buzz) in Prolog requires manipulating the logic such that the inference steps involved produce the desired control flow, which is obviously more difficult than specifying the control flow directly.

# Reflection

Prolog’s focus on logical programming violates the structured programming principle, reducing the readability, writability, and reliability of the program. The control flow of the program is decided by the inference rules, largely hidden from the programmer. As a result, the relationship between the rules written and the orders in which they are executed is difficult to determine. Complicating the issue are constructs such as the cut.

The interpreted and symbolic nature of Prolog decreases the reliability of code. Symbols/names are not resolved until runtime, meaning any errors involving invalid or nonexistent symbols will not be detected until they are used at runtime (possibly not detected at all if inference happens to not require them). Additionally, the usage of some symbols is restricted in some contexts (for example, arithmetic/numerical operations require the symbols to be “instantiated”), errors relating to which also cannot be detected until runtime.