Homework – LL(1) Parser

Goal:

Complete a LL(1) parser and implement new productions for a language agreed upon in class.

Purposes:

- Practice implementing another's work described in text (often much more difficult than it would seem at first glance).
- Gain experience implementing a compiler based on theory and not one that's ad-hoc.
- Gain experience creating valid productions for an LL(1) parser.

Instructions:

These instructions are given in the order you should complete them.

Implement the book's solution:

- Implementing a data structure to hold all the productions given in Figure 3.4 on page 101. Most solutions are some kind of collection of collections. Each non-terminal and terminal is usually identified through a key-value associative array or an index/value lookup array. Then each production is described as sequence of such keys or indexes.
- Implement the code to create the FIRST, FOLLOW, and $FIRST^+$ sets. These are described in the section Backtrack-Free Parsing (page 103), and are covered in Figures 3.7, 3.8, and the text. Note that when you have FIRST, you can build FOLLOW, and once those two are built, you can build $FIRST^+$. Please reach out if you struggle to follow the book's syntax here, and I will assist in explaining it to you.
    - Test that your code correctly creates each set before moving onto the next set.
- Implement the table-construction on Figure 3.12 on page 113. Make sure the table you generate matches the table on page 112. Note that the table's integer values refer to the index values in Figure 3.4 on page 101.
- Implement the pre-defined the LL(1) skeleton parser found in Figure 3.11 on page 112.
    - This algorithm utilizes a NextWord() function. You may use whatever solution you want for your NextWord(), whether it be ad-hoc or fully derived from insights in Chapter 2.
- Verify acceptance for all inputs in ll1_valid_book.txt and rejection for those in ll1_invalid_book.txt. An example of how an expression should parse is found in Figure 3.3 on page 114.

Add to the language:

- Create more productions to accept all expressions in ll1_valid_class.txt. A space cannot be part of a terminal.
- Verify that all expressions in ll1_invalid_book.txt are rejected.

Verification:

- Submit your code file, and any instructions how to compile it and then use it.
- Submit a PDF listing your productions.
- Also submit a short video (one minute or less) demonstrating accepting all expressions in ll1_valid_class.txt and rejecting those in ll1_invalid_book.txt.