CSCI-GA.2130-001 – Compiler Construction, Spring 2019

Project 1

Name: Yusen Su
NetID: ys3547
University ID (UID): N15412725
Collaborator Name: Yijian Xie
NetID: yx1891

Work Log

1. Project plan
    - Read the requirement and remarked the implementation hard point
    - Started generating comments, identifier and other tokens
    - Defined expression with precedence and associative required by definition
    - Defined type and statement
    - Defined declaration and program
    - Debugged and fixed issues of the parser
    - Addressed main, dangling else and other issues
    - Fixed part of those issues
    - Wrote the document
2. Project implementation
    - Using hacs and referenced by example files
    - Discussed issues with partners
        - i. How to write a correct regex for comments
        - ii. How to solve main function by parser
        - iii. How to solve dangling else

Remark

1. Fix comments
    - Single comment --- just allowed any other characters except newlines (\n).
    - Block comment --- not allowed comments nested. Each block comment only matches the closed end (*/).
        - E.g. /* ... /* ... */ */
            - */ are not match
2. Fix dangling else
    - I was trying to follow the open/close statements to fix dangling else, but the hacs will not allowed due to unreachable statement. Based on my understanding and discussed with my partner, we concluded the reason is that hacs does not allow open and close statements share the prefix (e.g. if <exp>).
    - I just implemented a transformation to avoid backtracking by using left factoring from class slides. I created a new sort, called IfTail, although, it still not solve the dangling else problem.
3. Precedence

- I just followed the hacs manual part 4 and several examples given. I defined each expression and followed those rules:
    - All unary operators are automatically using right recursive production with a same precedence expression.
    - The left recursive productions remain same as before, because hacs has a mechanism to eliminate it. The precedence of this is using the same way such as defining addition expression:
        - ⟦ ⟨Exp@5⟩ + ⟨Exp@6⟩ ⟧@5
    - Do not allow the recursion for comparison productions, just given the following precedence of the expressions.

4. Octal digits
    - I have discussed this and asked the format of this digit with TA. I implemented this feature by the octal representation in C string, that is, I allowed a string contains "\" followed by one to three octal digits.
5. Hexadecimal digits
    - Started an "\x" followed by two hexadecimal digits.
6. L-value
    - The manual requires L-value for each assignment expression. I implemented a new sort, called Lval, to handle this feature, which only allows an identifier or pointer dereference.
7. Main
    - The requirement needs a program with at least one defining a function main. However, this is not implemented in this parser.
        - The reason for that is the Hacs has limitation of productions, the different productions do not share the same prefix. So, there could only handle one (normal function) or the other (main function) each time if we want to hold this feature.
        - Thus, I left this requirement and considered the main function as a normal function.

<u>Testing</u>

1. Test performance
    - The following MC files will parse succeed
        - comments.MC, nomain.badMC, strcpy.MC, strings,MC, and strlen.MC
    - The following MC files will parse failed
        - comments.badMC, doubleelse.badMC, and nonlvalue.badMC
2. Test limitation
    - nomain.badMC
        - The parser will accept this file due to limit of the main function checking

<u>Code</u>

Please find Pr1Name.hx file for more details.