Student ID: 18249833
Name: Xhien Yi Tan (Xavier)

Programming Languages
Assignment Semester 2, 2016
Student ID: 18249833
Name: Xhien Yi Tan (Xavier)

This submission includes the following files:

- PL_EBNF.pdf
- PL_REPORT.pdf
- Source code (lexer.l, parser.y, Makefile)
    - Makefile has:
        - PL2016_check - as the main syntax checker
        - clean – delete all the compiled and generated files

The EBNF Grammar

PL_EBNF.pdf is derived from the syntax graphs that is provided from the Assignment Specification and this file follows the format from the Lecture Notes. Instead of putting everything on one single line, I have made it line after line to improve readability. In the file, I have created some custom rules to reduce repetition.

<constant_declaration_part>

- It represents <ident>"="<number>, and it is used for more than once in <constant_declaration>

<variable_declaration_part>

- It represents <ident>":"<ident>, and it is also used for more than once in <variable_declaration>

<semi_colon_and_statement>

- It represents a semi colon-separated of statement that is used in <compound_statement>

<add_subtract_term>

- It represents as add or subtract symbol and <term> that is used more than once in <expression>

<multiply_divide_id_num>

- It represents as multiply or divide symbol and <id_num> that is used more than once in <term>

<digit>

- It represents as a single integer number, and it is used in <number> so that <number> can have more than one digit.

<alphabet>

- It represents as a single alphabet, and it is used in <ident> so that <ident> can have more than one alphabet and make it identifier.

Lex (lexer.l)

I have defined 38 tokens for this lexer.l. This lexer file works by returning any token that has encountered to yacc for further operation. It also ignores white space and tab and also new line. It has fall back function whenever something goes wrong, it prints the unexpected character to the console and exit safely.

Parser (parser.y)

Yacc file has the following:

- 38 tokens which are the same as Lex file
- It has a function called found() which prints symbol that has been completely parsed
- It prints error whenever a syntax error is encountered using the function yyerror()
- It has additional rules to simplify the parser

Additional rules for **optional** statement:

optional_const_and_constant_declaration,
optional_var_and_var_declaration,
optional_type_declaration,
optional_procedure_interface,
optional_function_interface,
optional_formal_parameters

Optional statement has the form of:

optional_statement :

{ }
|
statement
{ }
;

Notes:
I have decided not to print anything in these optional statement is because theses optional statements are like wrapper function which calls the particular statement. It only prints when the actual statement is finished which is usually in the statement's action. For example, <optional_statement> calls <statement> and it prints message in <statement>.

Student ID: 18249833
Name: Xhien Yi Tan (Xavier)

statement :

TOKEN_A
TOKEN_B
{ found(“statement”); }
;

Additional rules for **repeated** statement:

constant_declaration_part1
constant_declaration_part2
variable_declaration_part1
variable_declaration_part2
semi_colon_and_ident
comma_and_ident
ident_and_semi_colon_and_block
semi_colon_and_statement
add_substract_term
multiply_divide_id_num

Repeated statement has the form of:

Repeated_statement :

statement
{ found("comma_and_ident"); }
|
repeated_statement
symbol
statement
{ found("comma_and_ident"); }
;

Notes:
Symbol can be a comma, semi colon, add, subtract, multiply or divide. The purpose of this is to reduce repetition.

Test file

Regarding the test file that is provided, one of the valid file named Implementation-Valid2.txt has fault in it. I have added a line “WHILE thing” before the “END DO” line to make it valid. I have tested my checker using the valid file and putting error in it for testing.