Experiment 8

Aim: Generate YACC specification for a few syntactic categories.

1. Program to recognize a valid arithmetic expression that uses operator $+$, $-$, * and $/$.
2. Program to recognize a valid variable which starts with a letter followed by any number of letters or digits.

To remember when using lex and yacc together

In the declaration section of lex program, include prog_name.tab.h for making lexer to read the input symbol, so that it can send tokens to Yacc parser. Also use -d switch while invoking bison.

1. Recognize a valid arithmetic expression

The lexical analyzer part should do the following:

- if input contains numbers, return NUM token to parser.
- if input contains a identifier, return ID token to parser.
- ignore tabspaces.
- if newline is encountered, it means end of expression, so return 0 to terminate lexer execution.
- any other character is returned as it is.

```
%%
[0-9]+(\.[0-9]+)?       { return NUM;}
[a-zA-Z_][_a-zA-Z0-9]*  { return ID; }
[\t]      ;
\n     return 0;
.    return yytext[0];
%%
```

1. Recognize a valid arithmetic expression

For the YACC part, we use the following grammar,
$E \rightarrow E + E | E * E | E - E | E / E | (E) | id | num$

- The terminal symbols *id* and *num* are the tokens recognized by lexical analyzer.
- The precedence and associativity of operators are also to be specified in the YACC specification.
- First define tokens which are getting returned from lexer, in the declaration section
- Then assign precedence and associativity of operators, first defined ones will have least preference. left means left-associativity, and right means right-associativity. , also in the declaration section
- The grammar productions are to be written in the rules section

2. Program to recognize a valid variable which starts with a letter followed by any number of letters or digits

We shall use the following grammar.

$stmt \rightarrow variable\ NL$

$variable \rightarrow LETTER\ alphanumeric$

$alphanumeric \rightarrow$
$LETTER\ alphanumeric | DIGIT\ alphanumeric | LETTER | DIGIT$

- Here, the tokens returned by lexer are newline, letters (which includes underscore) and digits.
- We must include prog_name.tab.c in our lexer.
- We can print valid identifier, when the expression is recognized, and invalid identifier in yyerror() function

Remember

Use the -d switch, while invoking bison to generate prog_name.tab.c