Practical No. 9

* Title: Parser for "For" Loop statements.

* Objective: Students will learn and implement

i) Parser for "For" Loop statements

ii) Rules sections for such a scanner and a parser and their working in synchronization.

* Description:

i) Create a LEX file first & create a RE for the digit and for the alphabet.

ii) In the rules section, write rules for number identification, identifier identification, FOR keyword, conditional ~~operations~~ operators like less than equal to, greater than equal to, equal to, not equal to, OR, AND, other details apart from this.

iii) Then create a YACC file. ~~include~~ In the declaration part of the YACC file include header file like stdio.h, stdlib.h.

iv) Declare ID, Num, for, LE, GE, EQ, NE, OR, AND as tokens. Declare precedence for considered tokens.

v) In the rules section of YACC file add rules for start symbol, definition, body, E and E2 which are in a way parts for "FOR" loop programming construct.

vi) Then include ~~the~~ lex.yy.c file which acts as a connector between LEX and YACC files. Lastly, call main routine for checking the FOR ~~of~~ expression through yyparse().

Code for LEX:

```
alpha [A-Za-z]
digit [0-9]

%%
[\t \n]
for                         return FOR;
{digit}+                    return NUM;
{alpha}({alpha}|{digit})*   return ID;
"<="                        return LE;
">="                        return GE;
"=="                        return EQ;
"!="                        return NE;
"||"                        return OR;
"&&"                        return AND;
.                           return yytext[0];
%%
```

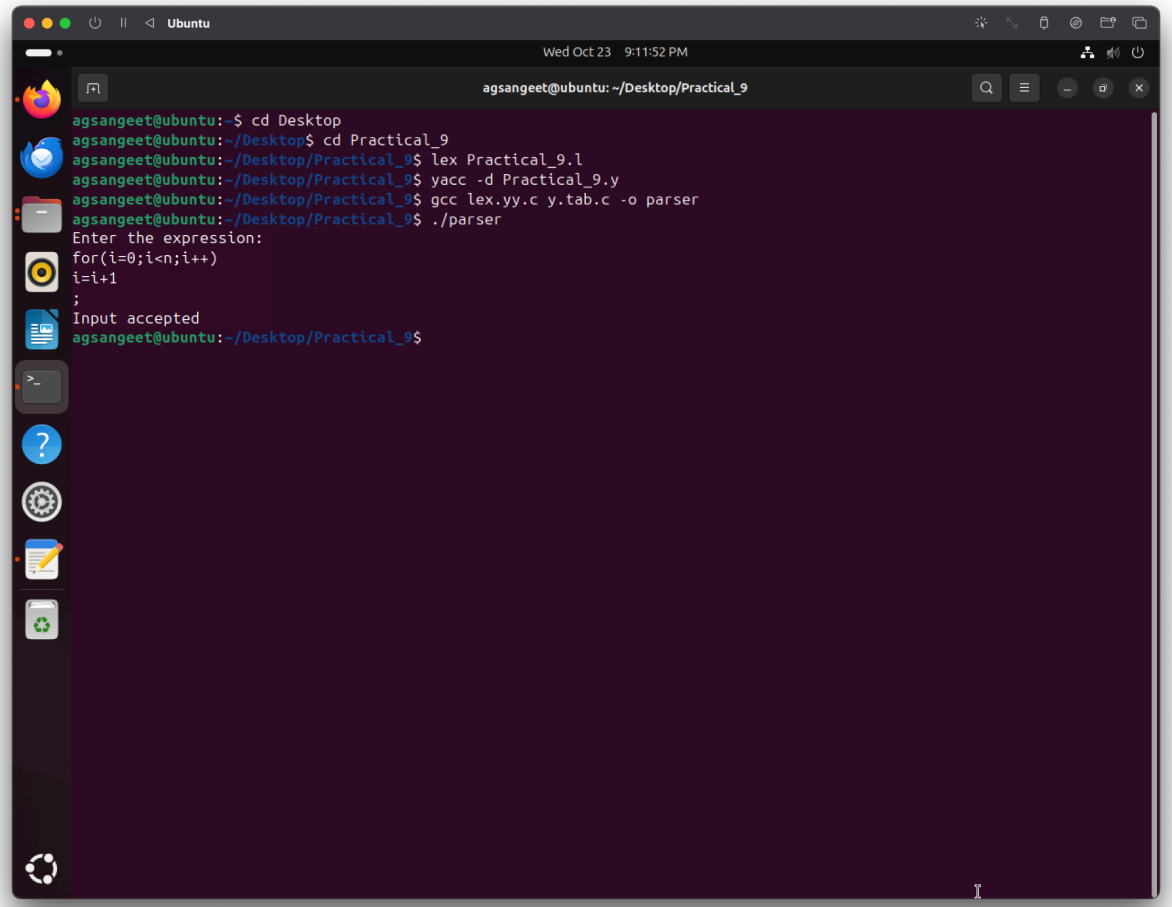
Code for YACC:

```
%{
#include <stdio.h>
#include <stdlib.h>

void yyerror(char*);
int yylex(void);

%}

%token ID NUM FOR LE GE EQ NE OR AND
%right "="
%left OR AND
%left '>' '<' LE GE EQ NE
%left '+' '-'
%left '*' '/'
%right UMINUS
%left '!'


%%
```

```
S          : ST {printf("Input accepted\n"); exit(0);}
ST        : FOR '(' E ';' E2 ';' E ')' DEF
            ;
DEF     : '{' BODY '}'
            | E';'
            | ST
            |
            ;
BODY   : BODY BODY
            | E ';'
            | ST
            |
            ;

E         : ID '=' E
           | E '+' E
           | E '-' E
           | E '*' E
           | E '/' E
           | E '<' E
           | E '>' E
           | E LE E
           | E GE E
           | E EQ E
           | E NE E
           | E OR E
           | E AND E
           | E '+' '+'
           | E '-' '-'
           | ID
           | NUM
           ;


E2      : E'<'E
          | E'>'E
          | E LE E
          | E GE E
          | E EQ E
          | E NE E
          | E OR E
```

```
        | E AND E
        ;
%%

#include "lex.yy.c"
int main() {
    printf("Enter the expression:\n");
    yyparse();
    return 0;
}

void yyerror(char* errorText){
    printf("%s",errorText);
}
```

Output:



* Conclusion: Thus, we have implemented the parser for "FOR" loop.