Nithin S
221IT085

IT250 Lab Assignment 7

Develop a parser using the LEX and YACC program, parse the given input to check either the “ Parsing is Successful ” or “ Parsing is Failed ”

1.a) “ if – else ” programming construct of "C" programming language

Lex Code 1.l

```
%{
#include "y.tab.h"
extern int yylval;
%}

%%

[\t \n]         /* Skip whitespace and newline characters */
if              { return IF; }
else            { return ELSE; }
[0-9]+          { return NUM; }
[A-Za-z_]+      { return ID; }
[char|short|int|long|float|double|bool|void|wchar_t|signed|unsigned] { return KEY; }
"<="            { return LE; }
">="            { return GE; }
"=="            { return EQ; }
"!="            { return NE; }
"||"            { return OR; }
"&&"            { return AND; }
">"             { return GT; }
"<"             { return LT; }
"//code"        { return STMT; }
.               { return yytext[0]; } /* Return any other character */
"$\n"           { return END; }      /* Return end of input */

%%

int yywrap() {
    return 1;
}
```

Yacc Code 1.y

```
%{
  #include<stdio.h>
  #include<stdlib.h>
  int yylex(void);
  int yyerror(const char *s);
  int success = 1;
%}

%token NUM ID LT GT EQ LE GE NE AND OR INC DEC END
%left '+' '-'
%left '*''/'
%right '^'
%right '='
%nonassoc UMINUS
%nonassoc IF
%nonassoc ELSE
%left GE NE LT GT LE EQ
%left AND OR

%%

S       : IF '(' F ')' '{' S '}' %prec IF
        | IF '(' F ')' '{' S '}' ELSE '{' S '}'
        | E ';'
        | E ';' S
        ;
F       : C LO C
        | C
        ;
LO      : AND
        | OR
        ;
C       : E RELOP E
        | E
        ;
E       : ID '=' E
        | E '+' E
        | E '-' E
        | E '*' E
        | E '/' E
        | E '^' E
        | '(' E ')'
        | '-' E %prec UMINUS
        | ID
        | NUM
        | ID INC
        | ID DEC
        ;
```

```
RELOP    :LT
         | GT
         | EQ
         | LE
         | GE
         | NE
         ;

%%

int main (void)
{
    yyparse();
    if(success)
        printf("Parsing is Successful\n");
    return 0;
}

int yyerror(const char *msg)
{
        printf("Parsing is Failed");
  success = 0;
        return 0;
}
```

Ouput

```
student@HP-Elite600G9-08:~/Desktop/assgn$ lex 1.l
student@HP-Elite600G9-08:~/Desktop/assgn$ yacc -d 1.y
1.y: warning: 69 shift/reduce conflicts [-Wconflicts-sr]
1.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
student@HP-Elite600G9-08:~/Desktop/assgn$ cc lex.yy.c y.tab.c
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

if (a>b)
{
   //code
}
else
{
   //code
}


$

Parsing is Successful

student@HP-Elite600G9-08:~/Desktop/assgn$
```

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

if (a>b)
{
   //code
}
else{

$

Parsing is Failed

student@HP-Elite600G9-08:~/Desktop/assgn$
```

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

else {
        //some code
}
Parsing is Failed
```

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

if {
        //code but no condition given
}
Parsing is Failed
```

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

if ( a < b) {
        //code
        if(1){
        //code
        }
}

$

Parsing is Successful
```

1.b “ switch case ” statements of "C" programming language

Code

Lex Code 1b.l

```
%option yylineno

%{
  #include "y.tab.h"
  extern int yylval;
%}

NUMBER        ([0-9]+(".")?([0-9])*)
IDENTIFIER    ([a-zA-z_][a-zA-z_0-9]*)
OP            (\+|\-|\*|\/|%|\-\-|\-|&&|\|\||>|<|==|>=|<=|=)

%%

[\t ]           /* ignore whitespaces */ ;

switch          {return SWITCH;}
case            {return CASE;}
break           {return BREAK;}
default         {return DEFAULT;}

{OP}            {return OP;}
{NUMBER}        {return NUM;}
{IDENTIFIER}    {return ID;}

.               {return yytext[0];}

\n              {yylval = yylineno;}

\n\n             {return 0;}

%%
```

Yacc Code 1b.l

```
%{
  #include<stdio.h>
  #include<stdlib.h>
  int yylex(void);
  int yyerror(const char *s);
  int success = 1;
%}

%token NUM ID SWITCH CASE BREAK DEFAULT OP
%left '+' '-'
%left '*''/'
%right '^'
%right '='
%nonassoc UMINUS

%%

S             : SWITCH '(' EXPR ')' '{' BODY  '}'
              ;
BODY          : BODY BODY
              | CASE EXPR ':' STMTS
              | CASE EXPR ':' STMTS BREAK ';'
              | CASE EXPR ':' STMTS BREAK ';' DEFAULT ':' STMTS
              ;
STMTS         : EXPR';'
              |
              ;
EXPR          : EXPR OP EXPR
              | ID
              | NUM
              |
              ;

%%

int main (void)
{
    yyparse();
    if(success)
        printf("Parsing is Successful\n");
    return 0;
}

int yyerror(const char *msg)
{
        printf("Parsing is Failed");
  success = 0;
        return 0;
}
```

OUTPUT

```
student@HP-Elite600G9-08:~/Desktop/assgn$ lex 1b.l
student@HP-Elite600G9-08:~/Desktop/assgn$ yacc -d 1b.y
1b.y: warning: 72 shift/reduce conflicts [-Wconflicts-sr]
1b.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
student@HP-Elite600G9-08:~/Desktop/assgn$ cc lex.yy.c y.tab.c
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

switch (day)
{
   case 1:
        //code
}
$

Parsing is Successful
```

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

switch (day)
{
   case 1:
        break;
   case 2:
        //code
        break;
   case 3:
        //code
   default:
        //code
}
$

Parsing is Successful
```

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

switch (day)
{
   case 1:
        //code
   case 2:
        //code
   case 3:
        //code
}
$

Parsing is Successful

student@HP-Elite600G9-08:~/Desktop/assgn$
```

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

switch (day)
{
   case 1
        break;
   case 2:
        //code
        break;
   case 3:
        //code
   default:
        //code
}

Parsing is Failed
```

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

switch()

Parsing is Failed
```

2a. “ for ’’ loop Construct

Lex Code 2a.l

```
%option yylineno

%{
  #include "y.tab.h"
  extern int yylval;
%}

NUMBER          ([0-9]+(".")?([0-9])*)
IDENTIFIER      ([a-zA-z_][a-zA-z_0-9]*)

%%

[\t ]             /* ignore whitespaces */ ;

for               {return FOR;}
{NUMBER}          {return NUM;}
{IDENTIFIER}      {return ID;}
"<="              {return LE;}
">="              {return GE;}
"=="              {return EQ;}
"!="              {return NE;}
"||"              {return OR;}
"&&"              {return AND;}

.                 {return yytext[0];}

\n                {yylval = yylineno;}

\n\n              {return 0;}

%%
```

Yacc Code 2a.y

```
%{
  #include<stdio.h>
  #include<stdlib.h>
  int yylex(void);
  int yyerror(const char *s);
  int success = 1;
%}

%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         : FOR '(' E ';' E2 ';' E ')' DEF
          ;
DEF       : '{' BODY '}'
          | E';'
          | S
          |
          ;
BODY      : BODY BODY
          | E ';'
          | S
          |
          ;
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
            ;

%%

int main (void)
{
    yyparse();
    if(success)
        printf("Parsing is Successful\n");
    return 0;
}

int yyerror(const char *msg)
{
        printf("Parsing Failed");
  success = 0;
        return 0;
}
```

Output

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

for(int i=0;i<10;i++)
{
        //code
}

$

Parsing is Successful
```

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

for(int i=0;i<10;i++)
{
        //code
        for(int j=0;j<55;j++)
        {
                //code
        }
}

$

Parsing is Successful
```

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

for()

Parsing is Failed

student@HP-Elite600G9-08:~/Desktop/assgn$
```

```
student@HP-Elite600G9-08:~/Desktop/assgn$ ./a.out

Enter the expression:

for(int i=0;i<10;i++)
{
        //code
        for(j++)
        {
                //code
        }
}
Parsing is Failed
```

2b “ while ’’ and “ do while ” loop Construct

Lex Code 2b.l

```
%option yylineno

%{
  #include "y.tab.h"
  extern int yylval;
%}

NUMBER        ([0-9]+(".")?([0-9])*)
IDENTIFIER    ([a-zA-z_][a-zA-z_0-9]*)

%%

[\t ]           /* ignore whitespaces */ ;

while           {return WHILE;}
{NUMBER}        {return NUM;}
{IDENTIFIER}    {return ID;}
"<="            {return LE;}
">="            {return GE;}
"=="            {return EQ;}
"!="            {return NE;}
"||"            {return OR;}
"&&"            {return AND;}

.               {return yytext[0];}

\n              {yylval = yylineno;}

\n\n            {return 0;}

%%
```

Yacc Code 2b.y

```
%{
  #include<stdio.h>
  #include<stdlib.h>
  int yylex(void);
  int yyerror(const char *s);
  int success = 1;
%}

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

%%

S         : WHILE '(' E2 ')' DEF
          ;
DEF       : '{' BODY '}'
          | E';'
          | S
          ;
BODY      : BODY BODY
          | E ';'
          | S
          |
          ;
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
            ;

%%

int main (void)
{
    yyparse();
    if(success)
        printf("Parsing is Successful\n");
    return 0;
}

int yyerror(const char *msg)
{
        printf("Parsing is Failed");
  success = 0;
}
```

Output

```
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$ lex 2b.l
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$ yacc -d 2b.y
2b.y: warning: 1 nonterminal useless in grammar [-Wother]
2b.y: warning: 1 rule useless in grammar [-Wother]
2b.y:62.1-4: warning: nonterminal useless in grammar: EXPR [-Wother]
   62 | EXPR : STMT
      | ^~~~
2b.y: warning: 71 shift/reduce conflicts [-Wconflicts-sr]
2b.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$ cc lex.yy.c y.tab.c
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$ ./a.out

Enter the expression:

while (a>b)
{
        //code
}

$

Parsing is Successful

nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$
```

```
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$ ./a.out

Enter the expression:

while (a>b)
{
        do
        {
           //code

        }while(1)
        //code
}
Parsing is Failed

nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$ }
```

```
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$ ./a.out

Enter the expression:

do
{
    //code
{while(1);

while(){}
Parsing is Failed
```

```
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$ ./a.out

Enter the expression:

while (a>b)
{
        do
        {
           //code

        }while(1);
        //code
}
$

Parsing is Successful
```

```
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$ ./a.out

Enter the expression:

do while ()

Parsing is Failed

nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_7$
```