Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester VI

5 March 2021

UCS1602 - Compiler Design

Exercise 5: Implementation of Desk Calculator using YaccTool

Objective:

Write Lex program to recognize relevant tokens required for the Yacc parser to implement desk calculator. Write the Grammar for the expression involving the operators. Precedence and associativity has to be preserved. Yacc is available as a command in linux. The grammar should have non-terminals E, Op and a terminal id.

Code:

Lex:

```
%{
    #include<stdio.h>
    #include "y.tab.c"
    extern YYSTYPE yylval;
%}

%%
[0-9]+ {yylval = atoi(yytext); return NUM;}
[\t] ;

[\n] return 0;

. return yytext[0];
%%
int yywrap(){
    return 1;
}
```

Yacc:

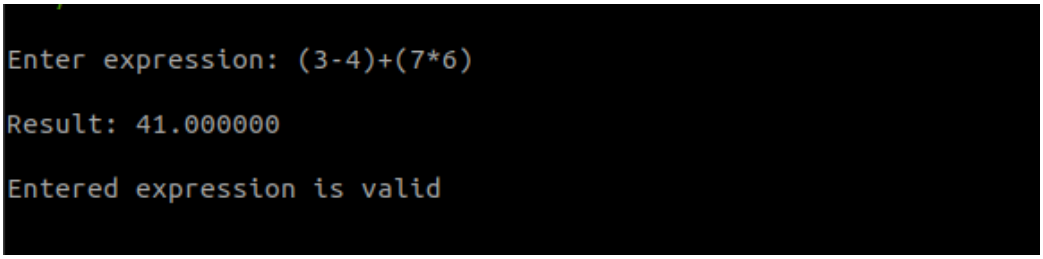
```
%{
    #include<stdio.h>
    #define YYSTYPE double
    int flag = 0;

    int yylex(void);

    double pow(double x, double y){
        double pdt = 1.0;
        while(y--){
            pdt *= x;
        }
        return pdt;
    }
    int op = 0;

%}

%token  NUM

%left '|'
%left '&'
```

```
%right '!'

%left '>' '<' '='

%left '+' '-'
%left '/' '*' '%'
%right '^'
%left '(' ')'

%%
P : E {printf("\nResult: %lf\n", $$);}
E : E '+' E {$$ = $1 + $3;}
  | E '-' E {$$ = $1 - $3;}
  | E '*' E {$$ = $1 * $3;}
  | E '/' E {$$ = $1 / $3;}
  | E '^' E {$$ = pow($1, $3);}
  |'('E')' {$$=$2;}
  | NUM {$$ = $1;}

E : E GR E {if($1 > $3){$$=1;} else{$$=0;}}
  | E GRE E {if($1 >= $3){$$=1;} else{$$=0;}}
  | E LE E {if($1 < $3){$$=1;} else{$$=0;}}
  | E LEE E {if($1 <= $3){$$=1;} else{$$=0;}}
  | E EQ E {if($1 == $3){$$=1;} else{$$=0;}}
  | E NEQ E {if($1 != $3){$$=1;} else{$$=0;}}

GR : '>'
GRE : '>''='
LE : '<'
LEE : '<''='
EQ : '=''='
NEQ : '!''='

E : E AND E {$$ = $1 * $3;}
  | E OR E {if($1==1||$3 ==1){$$=1;}else{$$=0;}}
  | NOT E { if($2==1){ $$=0; }else{ $$=1;}}

AND : '&''&'
OR : '|''|'
NOT : '!'

E : E LSHIFT E {$$ = (int)$1 << (int)$3;}
  | E RSHIFT E {$$ = (int)$1 >> (int)$3;}
  | E BAND E {$$ = (int)$1 & (int)$3;}
  | E BOR E {$$ = (int)$1 | (int)$3;}
```

```
68   | BNOT E {$$ = ~(int)$1;}
69
70 LSHIFT : '<''<'
71 RSHIFT : '>''>'
72 BAND : '&'
73 BOR : '|'
74 BNOT : '~'
75
76
77 ;
78 %%
79 int yyerror ()
80 {
81     printf("\nEntered expression is invalid\n\n");
82    flag=1;
83
84 }
85
86 int main (void){
87     printf("\nEnter expression: ");
88     yyparse();
89     if(flag==0)
90         printf("\nEntered expression is valid\n\n");
91     return 0;
92 }
```

Output:

Arithmetic Expression:



Boolean Expression:

```
Enter expression: 25==25

Result: 1.000000

Entered expression is valid
```

```
Enter expression: 25<=27

Result: 1.000000

Entered expression is valid
```

Bitwise operation:

```
Enter expression: 9<<1

Result: 18.000000

Entered expression is valid
```

```
Enter expression: 5&9

Result: 1.000000

Entered expression is valid
```

Learning Outcomes:

- Understood the basic working of Yacc tool.
- Learnt how to specify grammar in yacc.
- Learnt to use yacc efficiently to to perform actions for each grammatical structure.
