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:

```
1  %{
2      #include <stdio.h>
3      #include "y.tab.c"
4      extern YYSTYPE yylval;
5  %}
6
7  %%
8  [0-9]+ {yylval = atoi(yytext); return NUM;}
9  [\t] ;
10
11 [\n] return 0;
12
13 . return yytext[0];
14 %%
15 int yywrap(){
16     return 1;
17 }
```

Yacc:

```
1  %{
2      #include <stdio.h>
3      #define YYSTYPE double
4      int flag = 0;
5
6      int yylex(void);
7
8      double pow(double x, double y){
9          double pdt = 1.0;
10         while(y--){
11             pdt *= x;
12         }
13         return pdt;
14     }
15     int op = 0;
16
17 %}
18
19 %token   NUM
20 %left '|'
21 %left '&'
22 %right '!'
```

```
23
24 %left '>' '<'
25 %left '+' '-'
26 %left '/' '*' '%'
27 %right '^'
28 %left '(' ')'
29
30 %%
31 P : E {printf("\nResult: %lf\n", $$);}
32 E : E '+' E {$$ = $1 + $3;}
33   | E '-' E {$$ = $1 - $3;}
34   | E '*' E {$$ = $1 * $3;}
35   | E '/' E {$$ = $1 / $3;}
36   | E '^' E {$$ = pow($1, $3);}
37   |'('E')' {$$=$2;}
38   | NUM {$$ = $1;}
39
40 E : E AND E {$$ = $1 * $3;}
41   | E OR E {if($1==1||$3 ==1){$$=1;}else{$$=0;}}
42   | NOT E { if($2==1){ $$=0; }else{ $$=1;}}
43
44 AND : '&''&'
45 OR : '|''|'
46 NOT : '!'
47
48 E : E LSHIFT E {$$ = (int)$1 << (int)$3;}
49   | E RSHIFT E {$$ = (int)$1 >> (int)$3;}
50   | E BAND E {$$ = (int)$1 & (int)$3;}
51   | E BOR E {$$ = (int)$1 | (int)$3;}
52   | BNOT E {$$ = ~(int)$1;}
53
54 LSHIFT : '<''<'
55 RSHIFT : '>''>'
56 BAND : '&'
57 BOR : '|'
58 BNOT : '~'
59
60 E : E GR E {if($1 > $3){$$=1;} else{$$=0;}}
61   | E GRE E {if($1 >= $3){$$=1;} else{$$=0;}}
62   | E LE E {if($1 < $3){$$=1;} else{$$=0;}}
63   | E LEE E {if($1 <= $3){$$=1;} else{$$=0;}}
64   | E EQ E {if($1 == $3){$$=1;} else{$$=0;}}
65   | E NEQ E {if($1 != $3){$$=1;} else{$$=0;}}
66
67 GR : '>'
```

```
68  GRE  :  '>''='
69  LE  :  '<'
70  LEE  :  '<''='
71  EQ  :  '=''='
72  NEQ  :  '!''='
73

74

75  ;
76  %%
77  int  yyerror  ()
78  {
79      printf("\nEntered  expression  is  invalid\n\n");
80      flag=1;
81
82  }
83
84  int  main  (void){
85      printf("\nEnter  expression:  ");
86      yyparse();
87      if(flag==0)
88          printf("\nEntered  expression  is  valid\n\n");
89      return  0;
90  }
```

Output:

Arithmetic Expression:


```
Enter expression: (3-4)+(7*6)

Result: 41.000000

Entered expression is valid
```

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
```