Exercise 5
Desk Calculator Using Yacc

Praveen Kumar R
312217104114

February 3, 2020

Aim

To write a lex program to tokenize the the given algebric expression and a yacc program to fetch the tokens from the input expression string and parse it using a the grammar to valuate the expression.

Grammer

The grammer to parse the algebraic expression is:

$$
\begin{aligned}
&program \rightarrow lineprogram \mid line \\
&line \rightarrow expr \\
&expr \rightarrow expr\ +\ mulex \mid expr\ -\ mulex|mulex \\
&mulex \rightarrow mulex\ *\ powex \mid mulex\ /\ powex \mid powex \\
&powex \rightarrow powex\ \hat{}\ term \mid term \\
&term \rightarrow (\ expr\ ) \mid INTEGER
\end{aligned}
$$

Lex Code

```
%{
#include <stdlib.h>
#include <stdio.h>
#include "y.tab.h"
void yyerror(char*);
extern int yylval;
%}
%%
[ \t]+ ;
```

```
[0-9]+ {yylval = atoi(yytext);
        return INTEGER;}
[-+*/^] {return *yytext;}
"("     {return *yytext;}
")"     {return *yytext;}
\n      {return *yytext;}
.       {char msg[25];
         sprintf(msg,"%s <%s>","invalid character",yytext);
         yyerror(msg);}
%%
```

YACC Code

```
%{
#include <stdlib.h>
#include <stdio.h>
  int yylex(void);
extern FILE *yyin;
#include "y.tab.h"

  int pow2(int a, int b){
    int prod = 1;
    for(int i = 0;i< b;i++)
      prod*=a;
    return prod;
  }
%}
%token INTEGER
%%
program: line program
     | line

line: expr '\n' { printf("%d\n",$1); }

expr:expr '+' mulex { $$ = $1 + $3; }
     | expr '-' mulex { $$ = $1 - $3; }
     | mulex { $$ = $1; }

mulex: mulex '*' powex { $$ = $1 * $3; }
     | mulex '/' powex { $$ = $1 / $3; }
     | powex { $$ = $1; }

powex:powex '^' term {$$ = pow2($1, $3);}
     | term {$$ = $1;}
```

```
term: '(' expr ')' { $$ = $2; }
     | INTEGER { $$ = $1; }
%%
void yyerror(char *s)
{
  fprintf(stderr,"%s\n",s);
  return;
}
yywrap()
{
  return(1);
}
int main(void)
{
  char inputFile[100];
  printf("Enter the input file: ");
  scanf("%s",inputFile);
  yyin = fopen(inputFile, "r");
  yyparse();
  return 0;
}
```

Sample Input & Output 1

```
praveen@praveen/CompilerDesign/LALRparser:~$ lex parser.l
praveen@praveen/CompilerDesign/LALRparser:~$ yacc -d parser.y
praveen@praveen/CompilerDesign/LALRparser:~$ gcc y.tab.c lex.yy.c
praveen@praveen/CompilerDesign/LALRparser:~$ ./a.out
```

```
Enter the input file: input.in
12
57
49
41
2
16
64
```