Exp No: 8 **Date:**

Name: Pavan Kumar Ch **Regd No: 22501A05E0**

Experiment – 8

Aim: a. Implement a Yacc program to evaluate a given arithmetic expression.

Program:

LEX Program:

```
%{
#include "y.tab.h"
extern int yylval;
%}
%%
[0-9]+ { yylval = atoi(yytext); return NUMBER; }
[a-zA-Z]+ { return ID; }
[ \t]+ ; /* For skipping whitespaces */
\n { return 0; }
. { return yytext[0]; }
%%

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

YACC Program:

```
%{
#include <stdio.h>
int yylex(void);
int yyerror(char* s);
%}
%token NUMBER ID
%left '+' '-'
%left '*' '/'
%%
E : T { printf("Result = %d\n", $$); return 0; }
```

Exp No: 8 **Date:**

Name: Pavan Kumar Ch **Regd No: 22501A05E0**

```
T : T '+' T { $$ = $1 + $3; }
  | T '-' T { $$ = $1 - $3; }
  | T '*' T { $$ = $1 * $3; }
  | T '/' T { $$ = $1 / $3; }
  | '-' NUMBER { $$ = -$2; }
  | '-' ID { $$ = -$2; }
  | '(' T ')' { $$ = $2; }
  | NUMBER { $$ = $1; }
  | ID { $$ = $1; };
%%
int main() {
        printf("Enter the expression\n");
        yyparse();
}
int yyerror(char* s) {
        printf("\nExpression is invalid\n");
}
```

Output:

```
/mnt/c/U/pavan/Doc/G/CompilerDesign/08/expression 〉main ?1  lex expression.l

/mnt/c/U/pavan/Doc/G/CompilerDesign/08/expression 〉main ?1  yacc -d expression.y

/mnt/c/U/pavan/Doc/G/CompilerDesign/08/expression 〉main ?1  gcc lex.yy.c y.tab.c -ll

/mnt/c/U/pavan/Doc/G/CompilerDesign/08/expression 〉main ?1  ./a.out
Enter the expression
7*(18*45)-17
Result = 5653
```

Conclusion: Yacc program to evaluate a given arithmetic expression has been implemented successfully.

Exp No: 8 **Date:**

Name: Pavan Kumar Ch **Regd No: 22501A05E0**

Aim: b. YACC program to implement a Calculator and recognize a valid Arithmetic expression

Program:

LEX Program:

```
%{
#include <stdio.h>
#include "y.tab.h"
extern int yylval;
%}
/* Rule Section */
%%
[0-9]+ {
        yylval = atoi(yytext);
        return NUMBER;
}
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap() {
        return 1;
}
```

YACC Program:

```
%{
#include <stdio.h>
int yylex(void);
int yyerror(char* s);
int flag = 0;
%}
```

Exp No: 8 **Date:**

Name: Pavan Kumar Ch **Regd No: 22501A05E0**

```
%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
ArithmeticExpression: E {
        printf("\nResult=%d\n", $$);
        return 0;
};
E: E '+' E { $$ = $1 + $3; }
 | E '-' E { $$ = $1 - $3; }
 | E '*' E { $$ = $1 * $3; }
 | E '/' E { $$ = $1 / $3; }
 | E '%' E { $$ = $1 % $3; }
 | '(' E ')' { $$ = $2; }
 | NUMBER { $$ = $1; }
;
%%
int main() {
        printf("\nEnter Any Arithmetic Expression which can have operations Addition,
Subtraction, Multiplication, Division, Modulus, and Round brackets:\n");
        yyparse();
        if(flag == 0)
        printf("\nEntered arithmetic expression is Valid\n\n");
}
int yyerror(char* s) {
        printf("\nEntered arithmetic expression is Invalid\n\n");
        flag = 1;
        return 0;
}
```

Exp No: 8 **Date:**

Name: Pavan Kumar Ch **Regd No: 22501A05E0**

Output:

Case-1:

```
/mnt/c/U/pavan/Doc/G/CompilerDesign/08/calculator 〉main ?1  yacc -d calc.y                                    ✓〈 17:48:
/mnt/c/U/pavan/Doc/G/CompilerDesign/08/calculator 〉main ?1  gcc lex.yy.c y.tab.c -ll                           ✓〈 17:48:
/mnt/c/U/pavan/Doc/G/CompilerDesign/08/calculator 〉main ?1  ./a.out                                            ✓〈 17:49:

Enter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Division, Modulus, and Round brackets:
10-7-6

Result=-3

Entered arithmetic expression is Valid
```

Case-2:

```
/mnt/c/U/pavan/Doc/G/CompilerDesign/08/calculator 〉main ?1  ./a.out                                    ✓〈 15s 〈 17:49

Enter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Division, Modulus, and Round brackets:
7-8+

Entered arithmetic expression is Invalid
```

Conclusion: YACC program to implement a Calculator and recognize a valid Arithmetic expression has been implemented successfully.