Ex No: 7

Date:

EVALUATE EXPRESSION THAT TAKES DIGITS, *, + USING LEX AND YACC

AIM:

To perform arithmetic operations that takes digits,*, + using lex and yacc.

ALGORITHM:

Lex (exp7.l):

1. Recognizes sequences of digits and returns the token NUMBER.
2. Ignores tabs and newlines.
3. Returns any other single character as itself.
4. Indicates the end of input with yywrap().

Yacc (exp7.y):

1. Includes headers and declares global variables.
2. Declares token NUMBER.
3. Defines operator precedence and associativity.
4. Defines grammar rules for arithmetic expressions.
5. Prints the result of the expression evaluation in the ArithmeticExpression rule.
6. Handles syntax errors with yyerror().
7. The main function, prompts for an arithmetic expression, parses it, and prints whether it's valid or not based on the presence of syntax errors.

PROGRAM:

exp7.l:

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

Roll Number: 210701118
Name: Keerthana J

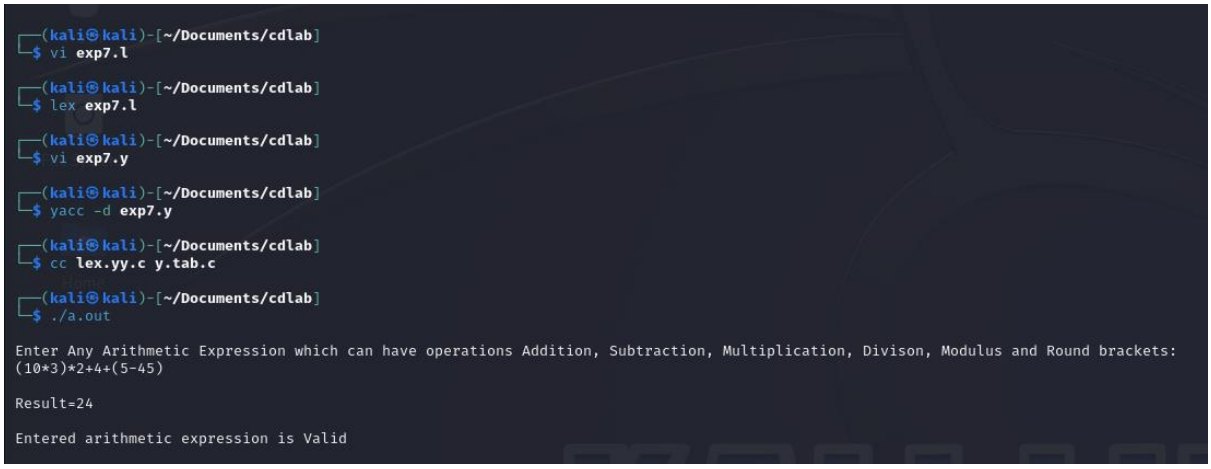
```
[0-9]+ {
        yylval=atoi(yytext);
        return NUMBER;
        }
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
return 1;
}
```

exp7.y:

```
%{
        #include<stdio.h>
        int flag=0;
        int yylex();
        void yyerror();
%}
%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;}
;
%%
void main(){
  printf("\nEnter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Divison, Modulus and Round brackets:\n");
  yyparse();
 if(flag==0)
  printf("\nEntered arithmetic expression is Valid\n\n");
}
void yyerror(){
  printf("\nEntered arithmetic expression is Invalid\n\n");
  flag=1;}
```

OUTPUT:



RESULT:

Thus, arithmetic operations that takes digits,*, + using lex and yacc have been performed.

Roll Number: 210701118
Name: Keerthana J

Ex No: 8

Date:

GENERATE THREE ADDRESS CODES

AIM:

To generate three address code using C program.

ALGORITHM:

- Get address code sequence.
- Determine current location of 3 using address (for 1st operand).
- If the current location does not already exist, generate move (B, O).
- Update address of A (for 2nd operand).
- If the current value of B and () is null, exist.
- If they generate operator () A, 3 ADPR.
- Store the move instruction in memory.

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void pm();
void plus();
void divi();
int i,ch,j,l,addr=100;
char ex[10], exp0[10] ,exp1[10],exp22[10],id1[5],op[5],id2[5];
char *strrev(char *str){
     char *p1, *p2;
     if (! str || ! *str)
          return str;
     for (p1 = str, p2 = str + strlen(str) - 1; p2 > p1; ++p1, --p2){
          *p1 ^= *p2;
          *p2 ^= *p1;
          *p1 ^= *p2;
     }
     return str;
}
void main(){
while(1){
printf("\n1.assignment\n2.arithmetic\n3.relational\n4.Exit\nEnter the choice:");
scanf("%d",&ch);
switch(ch){
case 1:
printf("\nEnter the expression with assignment operator:");
scanf("%s",exp0);
l=strlen(exp0);
exp22[0]='\0';
```

Roll Number: 210701118
Name: Keerthana J