**Assignment 4**

**Aim :**

Write a program to implement a parser in the YACC tool

4.A To evaluate an arithmetic expression

4.B To evaluate or check built-in

4.C To recognize the valid variable name

**Theory :**

For Compiling YACC Program:

1. Write lex program in a file file.l and yacc in a file file.y
2. Open Terminal and navigate to the Directory where you have saved the files.
3. type lex file.l
4. type yacc file.y
5. type cc lex.yy.c y.tab.h -ll
6. type ./a.out

YACC: -

A parser generator is a program that takes as input a specification of

syntax and produces as output a procedure for recognizing that

language. Historically, they are also called compiler-compilers.

YACC (yet another compiler-compiler) is an LALR(1) (LookAhead, Left-to-

right, Rightmost derivation producer with 1 lookahead token) parser

generator. YACC was originally designed for being complemented by Lex.

**Code :**

expression.l

%{

#include<stdio.h>

#include "y.tab.h"

extern int yylval;

%}

%%

[0-9]+ {

yylval=atoi(yytext);

return NUMBER;

}

[\t] ;

[\n] return 0;

. return yytext[0];

%%

int yywrap()

{

return 1;

}

expression.y

%{

#include<stdio.h>

%}

%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("Enter the Expression ");

yyparse();

return 0;

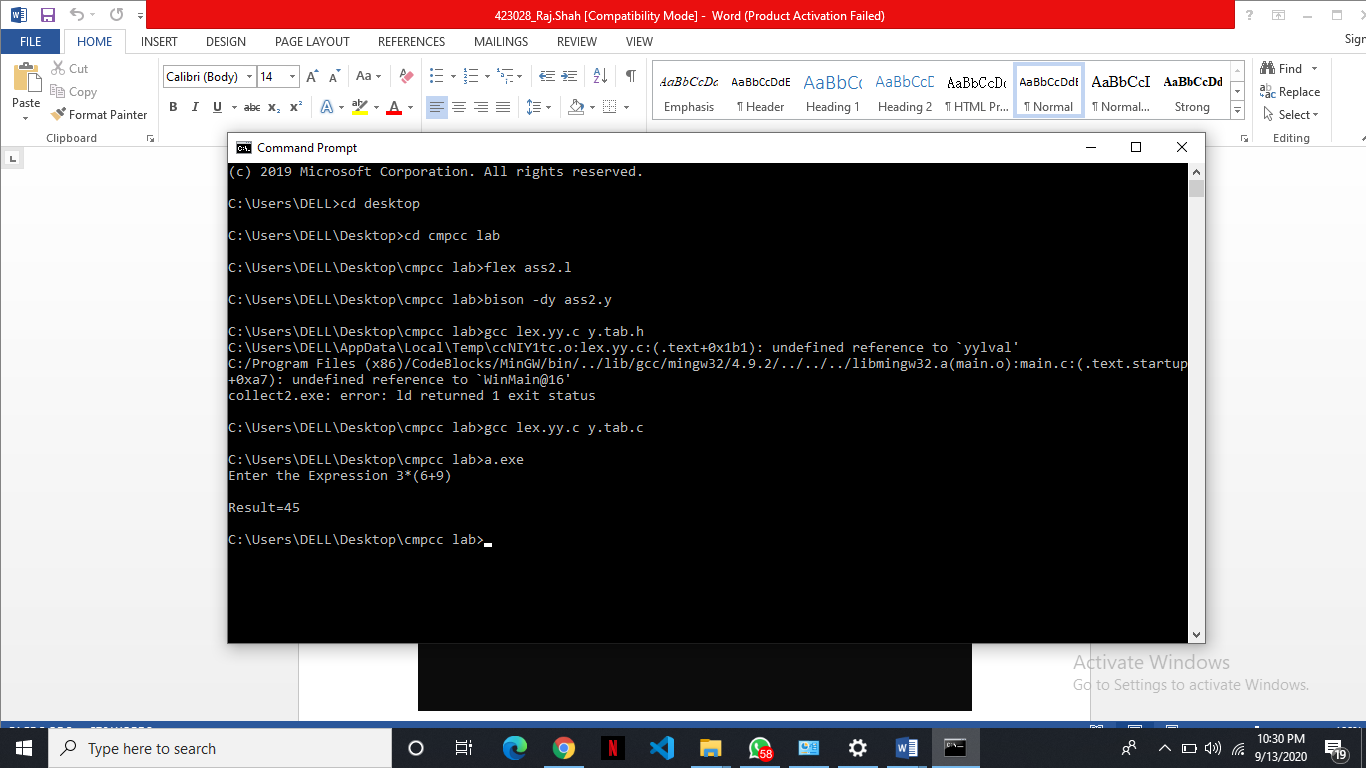
}

int yyerror(char\* s) {

printf("\nExpression is invalid\n");

}

**Output:**



**Conclusion:**

Lex analyzer and parser were implemented to evaluate the expression.