Code(.l):  
%{

#include "y.tab.h"

%}

alpha [A-Za-z]

digit [0-9]

%%

[ \t\n]

if return IF;

then return THEN;

else return ELSE;

while return WHILE;

for return FOR;

do return DO;

{digit}+ return NUM;

{alpha}({alpha}|{digit})\* return ID;

"<=" return LE;

">=" return GE;

"==" return EQ;

"!=" return NE;

"||" return OR;

"&&" return AND;

. return yytext[0];

%%

int yywrap(){

return 1;

}

Code(.y):  
%{

#include <stdio.h>

#include <stdlib.h>

%}

%token ID NUM IF THEN ELSE WHILE FOR DO LE GE EQ NE OR AND

%right '='

%left AND OR

%left '<' '>' LE GE EQ NE

%left '+''-'

%left '\*''/'

%right UMINUS

%left '!'

%%

S : ST { printf("Input accepted.\n"); exit(0); };

ST : IF '(' E2 ')' THEN ST1 ';'

| IF '(' E2 ')' THEN ST1 ';' ELSE ST1 ';'

| WHILE '(' E2 ')' ST1 ';'

| DO ST WHILE '(' E2 ')' ';'

| FOR '(' E2 ';' E2 ';' E2 ')' ST ';'

;

ST1 : ST

| E

;

E : ID '=' E

| E '+' E

| E '-' E

| E '\*' E

| E '/' E

| E '<' E

| E '>' E

| E LE E

| E GE E

| E EQ E

| E NE E

| E OR E

| E AND E

| ID

| NUM

;

E2 : E'<'E

| E'>'E

| E LE E

| E GE E

| E EQ E

| E NE E

| E OR E

| E AND E

| ID

| NUM

;

%%

main()

{

printf("Enter the exp: ");

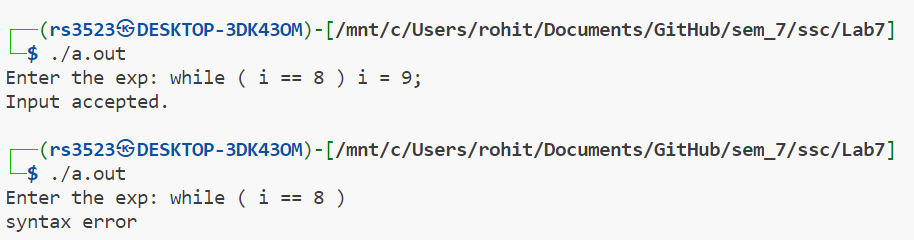
yyparse();

}

void yyerror(char const \*s) {

fprintf(stderr, "%s\n", s);

}

Output:  


Code(.l):

%{

/\* Definition section \*/

#include<stdlib.h>

#include "y.tab.h"

extern int yylval;

%}

/\* Rule Section \*/

%%

[0-9]+ {

yylval=atoi(yytext);

return NUMBER;

}

'<=' return LE;

'>=' return GE;

'!=' return NE;

'==' return EQ;

[\t] ;

[\n] return 0;

. return yytext[0];

%%  
  
Code(.y):  
%{

/\* Definition section \*/

#include<stdio.h>

int flag=0;

%}

%token NAME NUMBER

%left GE LE EQ NE EE '<' '>'

%left '+' '-'

%left '\*' '/' '%'

%left '(' ')'

%nonassoc UMINUS

/\* Rule Section \*/

%%

ArithmeticExpression: E{

printf("Result=%d", $$);

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;}

|E GE E {$$=$1 >= $3 ;}

|E LE E {$$=$1 <= $3 ;}

|E NE E {$$=$1 != $3 ;}

|E EE E {$$=$1 == $3 ;}

|UMINUS E {$$=-$1 ;}

;

%%

//driver code

int main()

{

//printf("\nEnter the Expression:\n");

yyparse();

//if(flag==0)

//printf("\nEntered arithmetic expression is Valid\n\n");

// return 0;

}

int yyerror()

{

// printf("\nEntered arithmetic expression is Invalid\n\n");

// flag=1;

}

int yywrap(){

return 1;

}

Output:

