11. **yacc** program for an advanced desk calculator

**YACC File (calc.y)**

%{

#include <stdio.h>

#include <stdlib.h>

void yyerror(const char \*s);

int yylex(void);

%}

%union {

int num;

}

%token <num> NUMBER

%left '+' '-'

%left '\*' '/'

%right UMINUS

%%

input:

/\* empty \*/

| input line

;

line:

'\n'

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

;

expr:

NUMBER { $$ = $1; }

| expr '+' expr { $$ = $1 + $3; }

| expr '-' expr { $$ = $1 - $3; }

| expr '\*' expr { $$ = $1 \* $3; }

| expr '/' expr { $$ = $1 / $3; }

| '-' expr UMINUS { $$ = -$2; }

| '(' expr ')' { $$ = $2; }

;

%%

int main(void) {

return yyparse();

}

void yyerror(const char \*s) {

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

}

**Lex File (calc.l)**

**%{**

**#include "y.tab.h"**

**%}**

**%%**

**[0-9]+ { yylval.num = atoi(yytext); return NUMBER; }**

**[ \t] { /\* ignore whitespace \*/ }**

**\n { return '\n'; }**

**. { return yytext[0]; }**

**%%**

**int yywrap(void) {**

**return 1;**

**}**

**Compilation and Execution**

1. **Generate the parser code**:

yacc -d calc.y

1. **Generate the lexical analyzer code**:

lex calc.l

1. **Compile the generated C code**:

cc y.tab.c lex.yy.c -o calc -ll

1. **Run the calculator**:

./calc