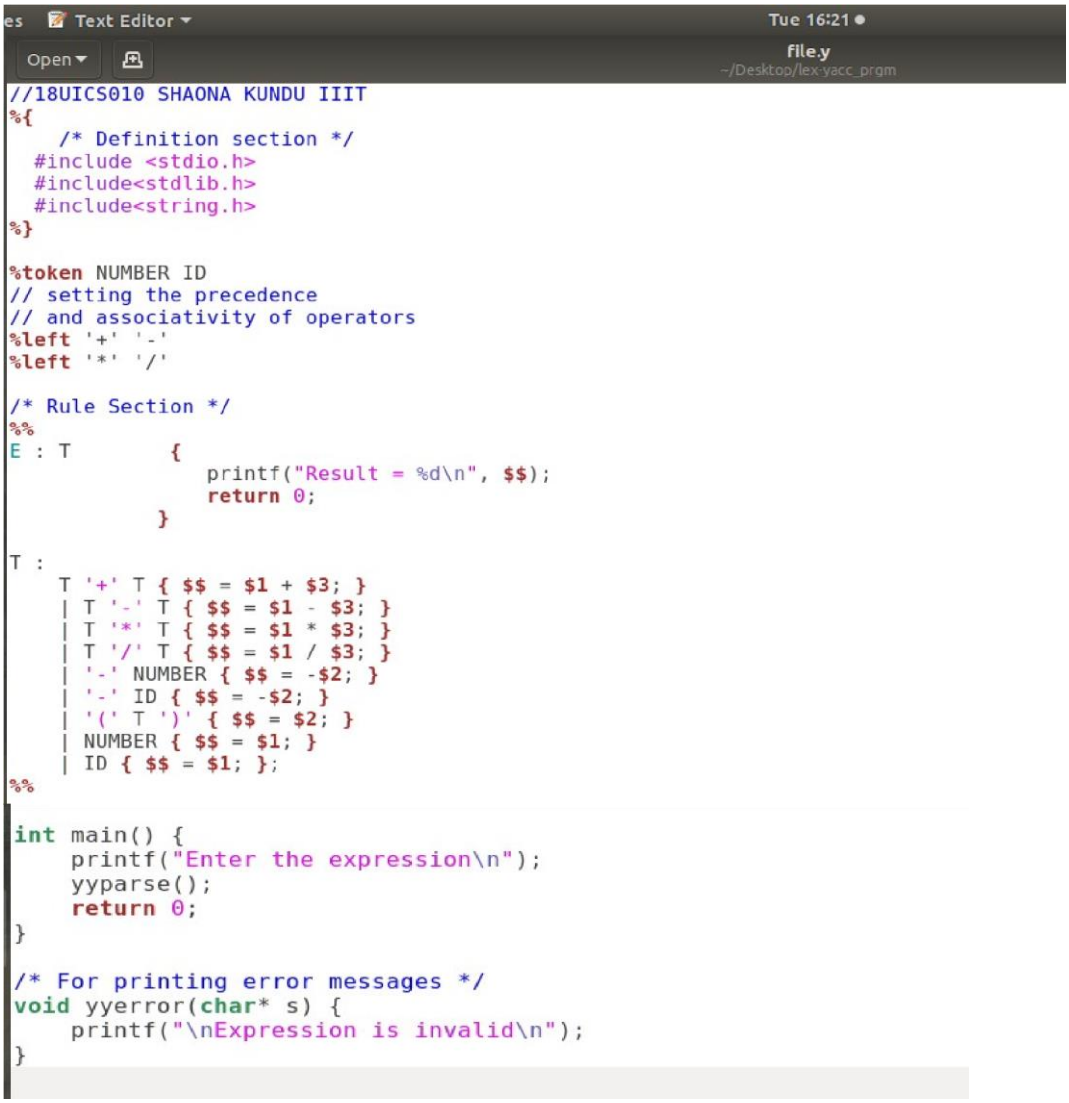
Output-

1. **Write a c program to implement program semantic rules to calculate the expression that takes an expression with digits, + and * and compute the value**

Input - Yacc

code-




```
//18UICS010 SHAONA KUNDU IIIT
%{
    /* Definition section */
  #include <stdio.h>
  #include<stdlib.h>
  #include<string.h>
%}

%token NUMBER ID
// setting the precedence
// and associativity of operators
%left '+' '-'
%left '*' '/'

/* Rule Section */
%%
E : T        {
                printf("Result = %d\n", $$);
                return 0;
             }

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();
    return 0;
}

/* For printing error messages */
void yyerror(char* s) {
    printf("\nExpression is invalid\n");
}
```

Lex code-

Text Editor Tue 16:21

Open file.l
~/Desktop/lex-yacc_prgm

```
//18UICS010 SHAONA KUNDU
%{
    /* Definition section*/
    #include "y.tab.h"
    extern yylval;
%}

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

[a-zA-Z]+    { return ID; }
[ \t]+         ;  /*For skipping whitespaces*/

\n            { return 0; }
.            { return yytext[0]; }

%%
```

Output -

```
(base) user@user-Vostro-15-3568:~/Desktop/lex-yacc_prgm$ lex file.l
(base) user@user-Vostro-15-3568:~/Desktop/lex-yacc_prgm$ yacc -d file.y
(base) user@user-Vostro-15-3568:~/Desktop/lex-yacc_prgm$ cc lex.yy.c y.tab.c -ll
(base) user@user-Vostro-15-3568:~/Desktop/lex-yacc_prgm$ ./a.out
Enter the expression
6*(4-2)/2
Result = 6
(base) user@user-Vostro-15-3568:~/Desktop/lex-yacc_prgm$ ./a.out
Enter the expression
19*19+10
Result = 371
(base) user@user-Vostro-15-3568:~/Desktop/lex-yacc_prgm$
```