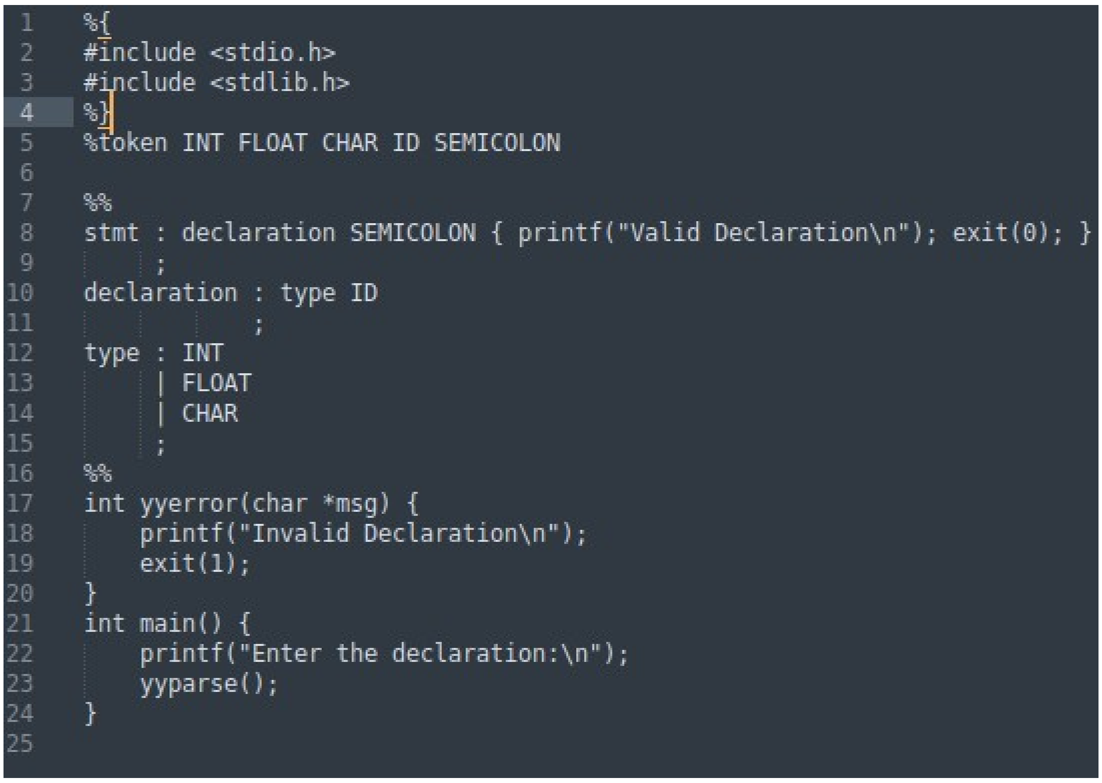
Lab 10

Name: AYANIKA PAUL
Reg no. 220905128
Roll No. 22

Q1)Write a bison program,
1. To check a valid declaration statement.

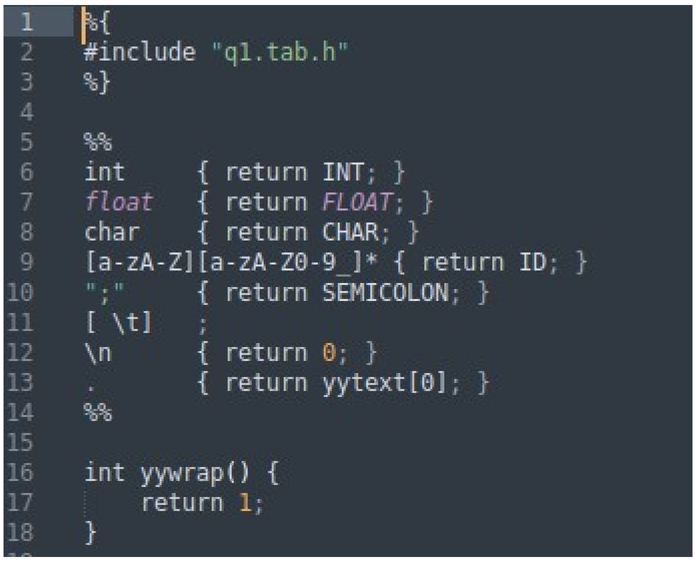
CODE:

q1.y



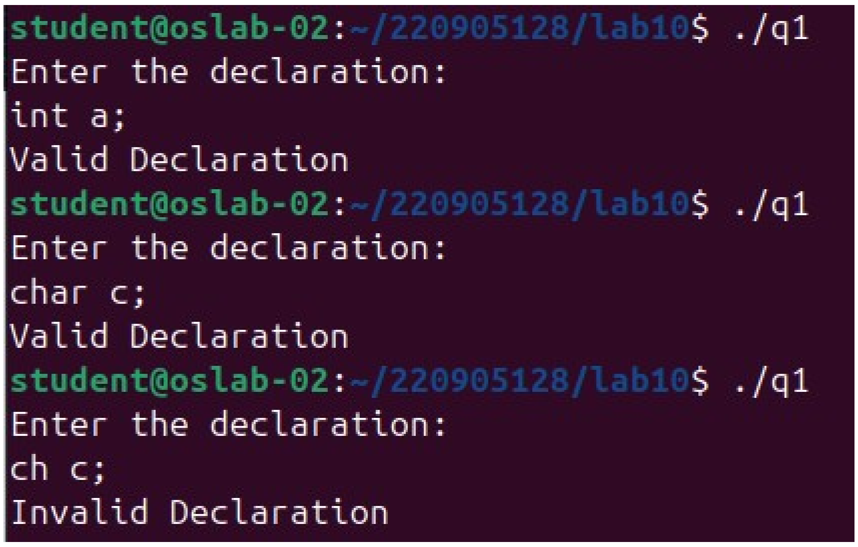
```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%token INT FLOAT CHAR ID SEMICOLON

%%
stmt : declaration SEMICOLON { printf("Valid Declaration\n"); exit(0); }
     ;
declaration : type ID
            ;
type : INT
     | FLOAT
     | CHAR
     ;
%%
int yyerror(char *msg) {
    printf("Invalid Declaration\n");
    exit(1);
}
int main() {
    printf("Enter the declaration:\n");
    yyparse();
}
```

q1.l



```
%{
#include "q1.tab.h"
%}

%%
int     { return INT; }
float   { return FLOAT; }
char    { return CHAR; }
[a-zA-Z][a-zA-Z0-9_]* { return ID; }
";"     { return SEMICOLON; }
[ \t]   ;
\n      { return 0; }
.       { return yytext[0]; }
%%

int yywrap() {
    return 1;
}
```

OUTPUT:

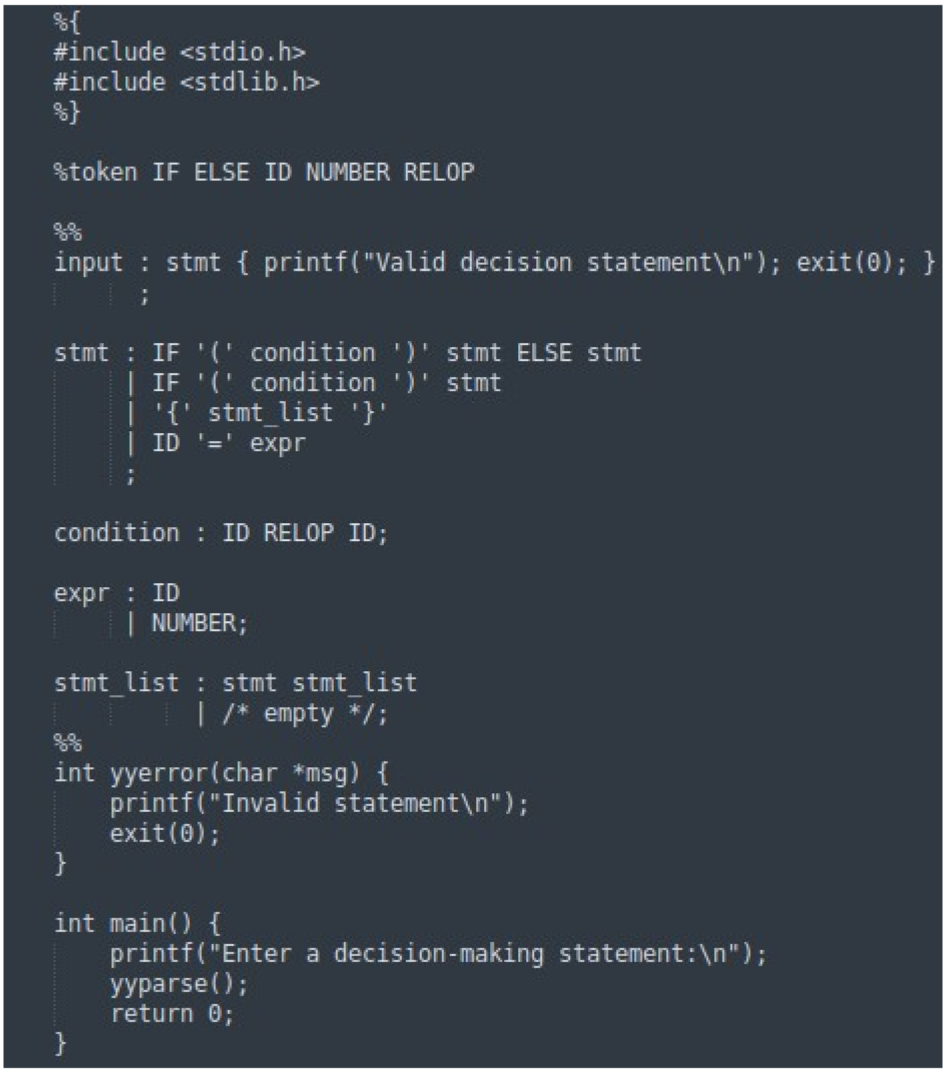


```
student@oslab-02:~/220905128/lab10$ ./q1
Enter the declaration:
int a;
Valid Declaration
student@oslab-02:~/220905128/lab10$ ./q1
Enter the declaration:
char c;
Valid Declaration
student@oslab-02:~/220905128/lab10$ ./q1
Enter the declaration:
ch c;
Invalid Declaration
```

2. To check a valid decision making statements.

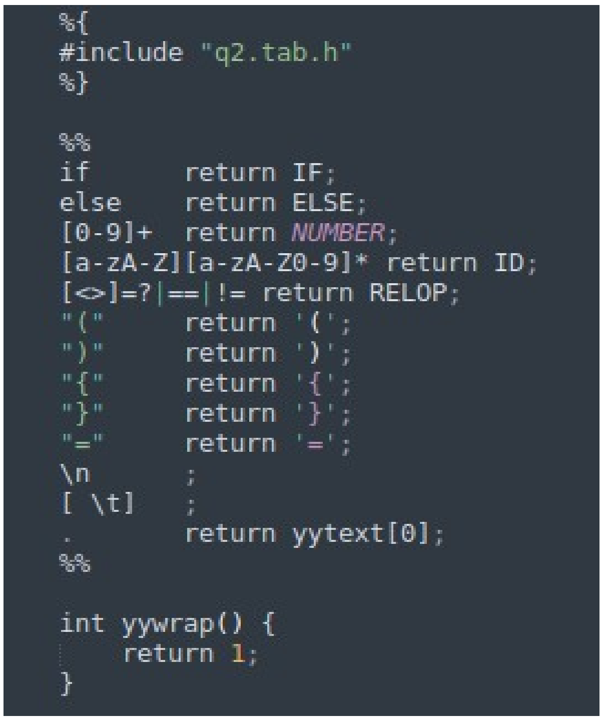
CODE:

q2.y



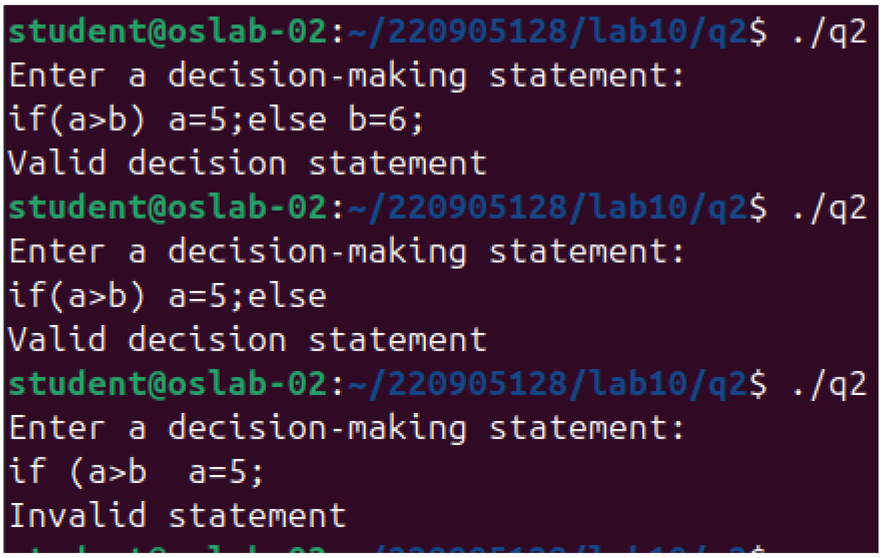
```
%{
#include <stdio.h>
#include <stdlib.h>
%}

%token IF ELSE ID NUMBER RELOP

%%
input : stmt { printf("Valid decision statement\n"); exit(0); }
      ;

stmt : IF '(' condition ')' stmt ELSE stmt
     | IF '(' condition ')' stmt
     | '{' stmt_list '}'
     | ID '=' expr
     ;

condition : ID RELOP ID;

expr : ID
     | NUMBER;

stmt_list : stmt stmt_list
          | /* empty */;
%%
int yyerror(char *msg) {
    printf("Invalid statement\n");
    exit(0);
}

int main() {
    printf("Enter a decision-making statement:\n");
    yyparse();
    return 0;
}
```

q2.l



```
%{
#include "q2.tab.h"
%}

%%
if      return IF;
else    return ELSE;
[0-9]+  return NUMBER;
[a-zA-Z][a-zA-Z0-9]* return ID;
[<>]=?|==|!= return RELOP;
"("     return '(';
")"     return ')';
"{"     return '{';
"}"     return '}';
"="     return '=';
\n      ;
[ \t]   ;
.       return yytext[0];
%%

int yywrap() {
    return 1;
}
```

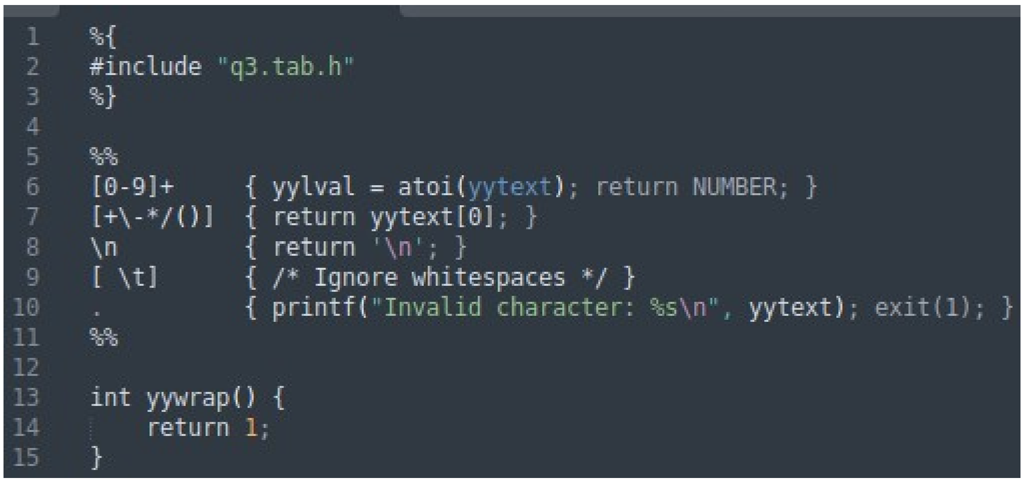
OUTPUT:



```
student@oslab-02:~/220905128/lab10/q2$ ./q2
Enter a decision-making statement:
if(a>b) a=5;else b=6;
Valid decision statement
student@oslab-02:~/220905128/lab10/q2$ ./q2
Enter a decision-making statement:
if(a>b) a=5;else
Valid decision statement
student@oslab-02:~/220905128/lab10/q2$ ./q2
Enter a decision-making statement:
if (a>b  a=5;
Invalid statement
```

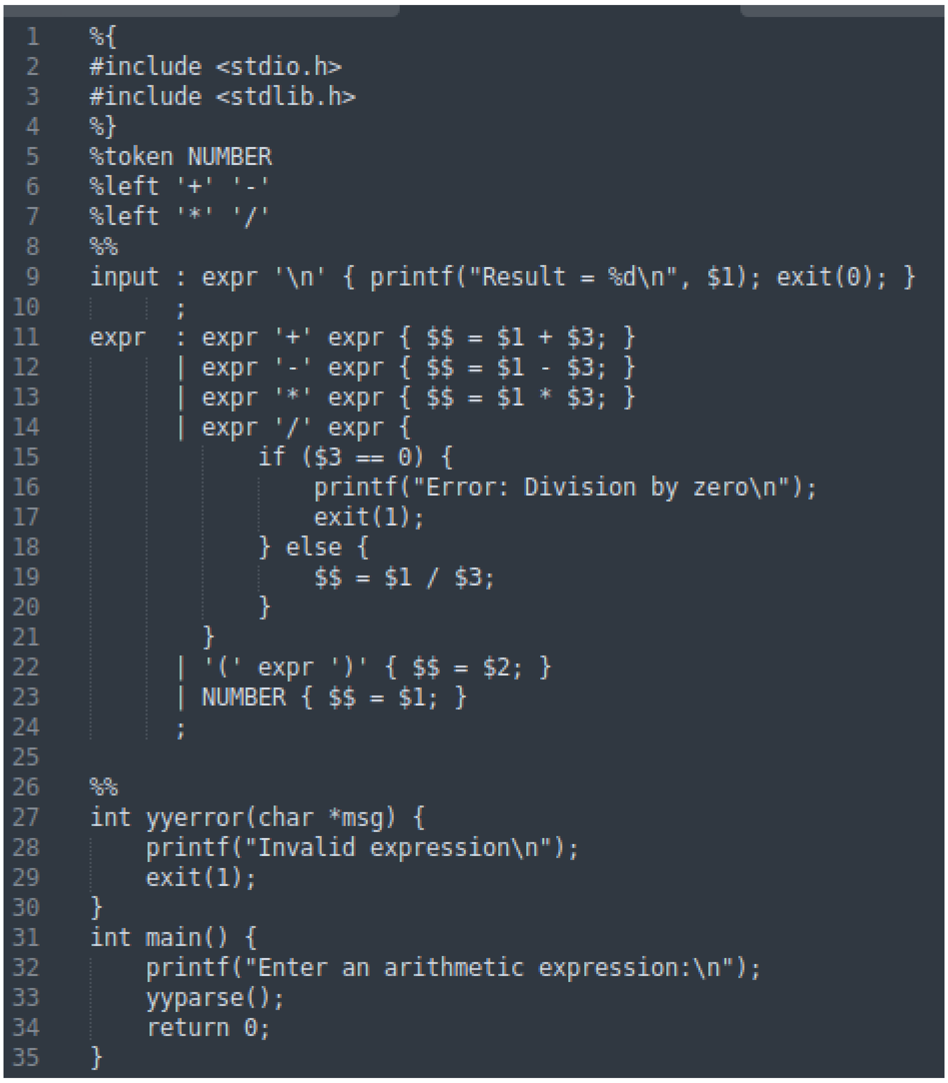
3. To evaluate an arithmetic expression involving operations +,-,* and /.

q3.l



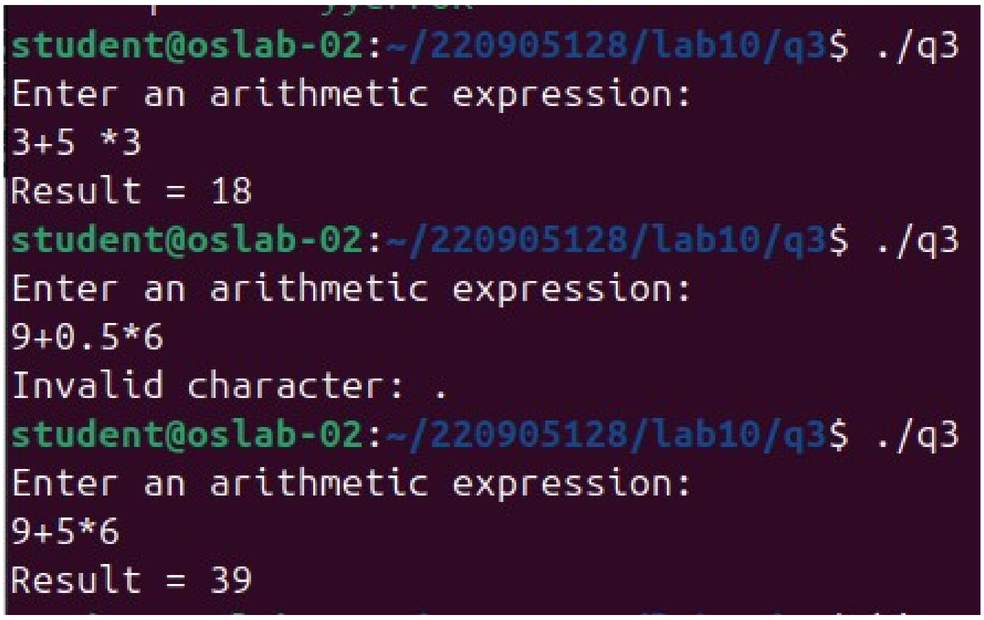
```
%{
#include "q3.tab.h"
%}

%%
[0-9]+     { yylval = atoi(yytext); return NUMBER; }
[+\-*/()]  { return yytext[0]; }
\n         { return '\n'; }
[ \t]      { /* Ignore whitespaces */ }
.          { printf("Invalid character: %s\n", yytext); exit(1); }
%%

int yywrap() {
    return 1;
}
```

q3.y



```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%token NUMBER
%left '+' '-'
%left '*' '/'
%%
input : expr '\n' { printf("Result = %d\n", $1); exit(0); }
      ;
expr  : expr '+' expr { $$ = $1 + $3; }
      | expr '-' expr { $$ = $1 - $3; }
      | expr '*' expr { $$ = $1 * $3; }
      | expr '/' expr {
            if ($3 == 0) {
                printf("Error: Division by zero\n");
                exit(1);
            } else {
                $$ = $1 / $3;
            }
        }
      | '(' expr ')' { $$ = $2; }
      | NUMBER { $$ = $1; }
      ;

%%
int yyerror(char *msg) {
    printf("Invalid expression\n");
    exit(1);
}
int main() {
    printf("Enter an arithmetic expression:\n");
    yyparse();
    return 0;
}
```

OUTPUT:

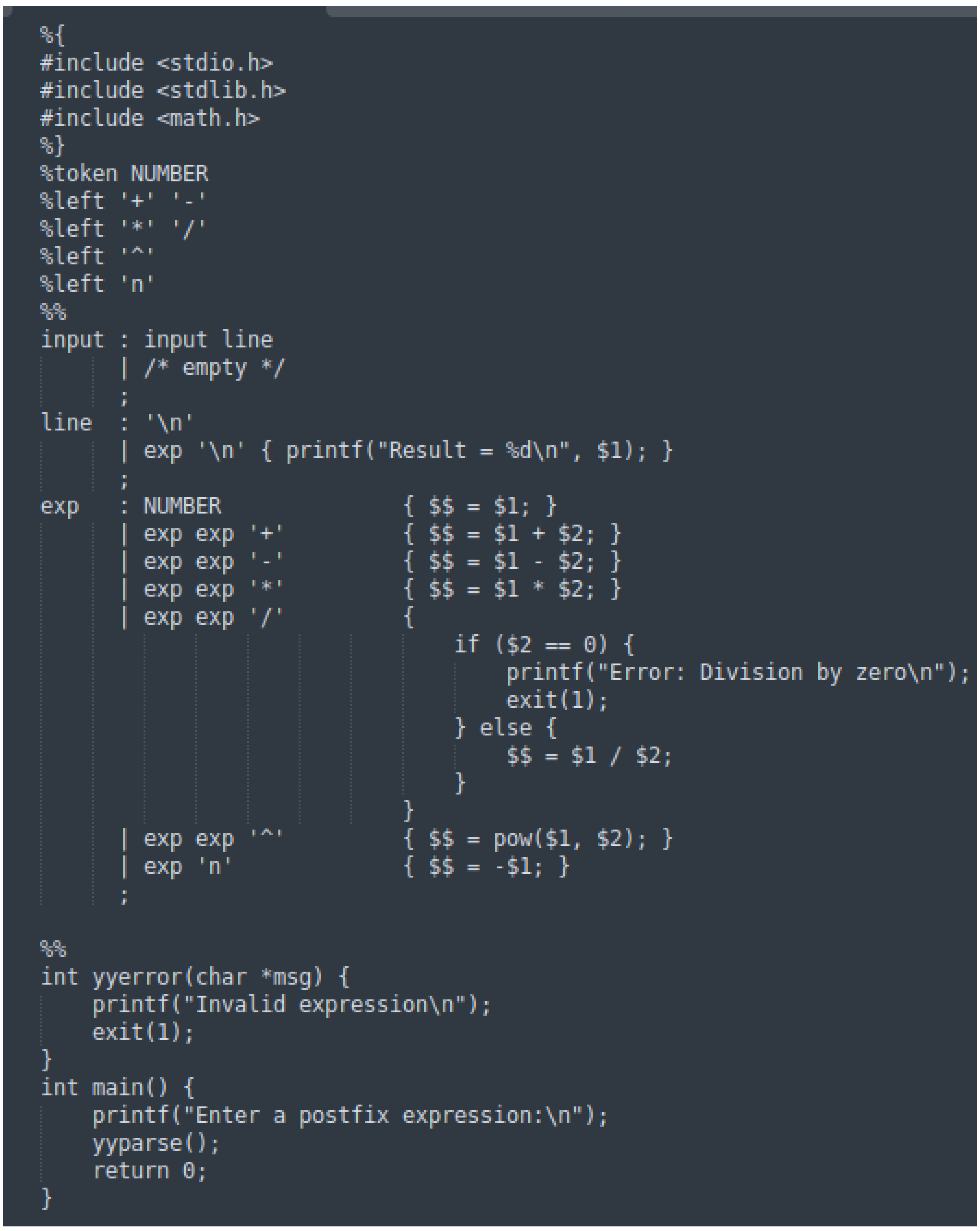


```
student@oslab-02:~/220905128/lab10/q3$ ./q3
Enter an arithmetic expression:
3+5 *3
Result = 18
student@oslab-02:~/220905128/lab10/q3$ ./q3
Enter an arithmetic expression:
9+0.5*6
Invalid character: .
student@oslab-02:~/220905128/lab10/q3$ ./q3
Enter an arithmetic expression:
9+5*6
Result = 39
```

4. To validate a simple calculator using postfix notation. The grammar rules are as follows –
input → input line | ε
line → ‘\n’ | exp ‘\n’
exp → num | exp exp ‘+’
| exp exp ‘-‘
| exp exp ‘*’
| exp exp ‘/’
| exp exp ‘^’
| exp ‘n’

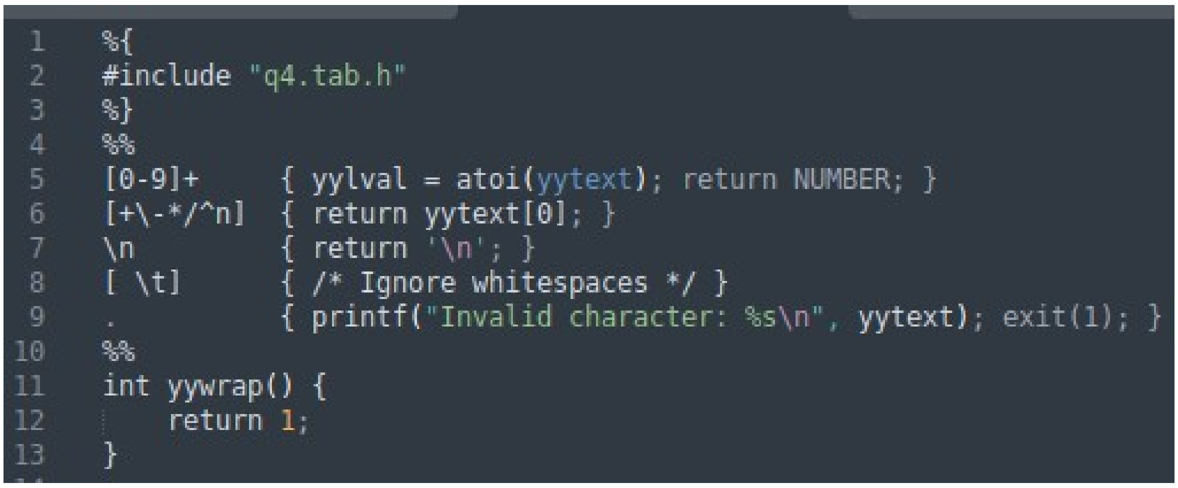
CODE:

q4.y



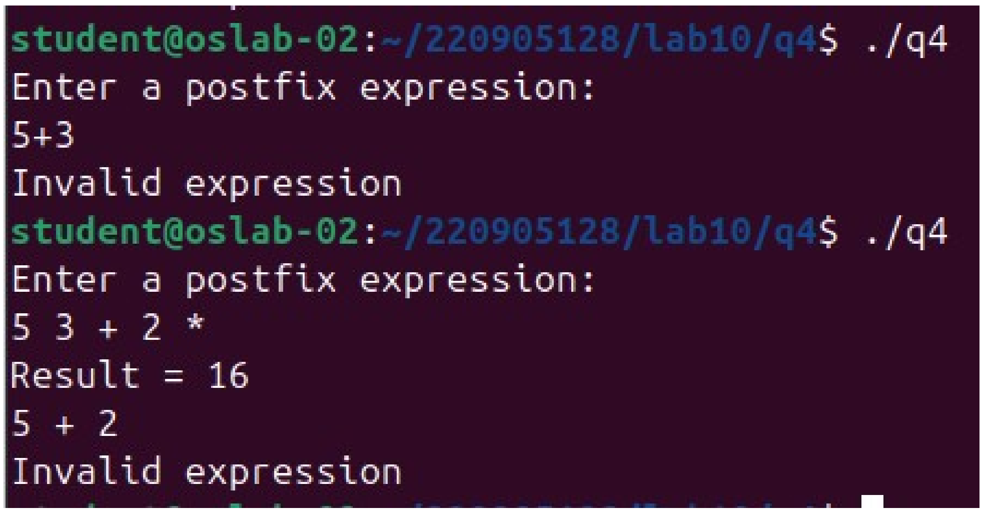
```
%{
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
%}
%token NUMBER
%left '+' '-'
%left '*' '/'
%left '^'
%left 'n'
%%
input : input line
      | /* empty */
      ;
line  : '\n'
      | exp '\n' { printf("Result = %d\n", $1); }
      ;
exp   : NUMBER              { $$ = $1; }
      | exp exp '+'         { $$ = $1 + $2; }
      | exp exp '-'         { $$ = $1 - $2; }
      | exp exp '*'         { $$ = $1 * $2; }
      | exp exp '/'         {
                                if ($2 == 0) {
                                    printf("Error: Division by zero\n");
                                    exit(1);
                                } else {
                                    $$ = $1 / $2;
                                }
                            }
      | exp exp '^'         { $$ = pow($1, $2); }
      | exp 'n'             { $$ = -$1; }
      ;

%%
int yyerror(char *msg) {
    printf("Invalid expression\n");
    exit(1);
}
int main() {
    printf("Enter a postfix expression:\n");
    yyparse();
    return 0;
}
```

q4.l



```
%{
#include "q4.tab.h"
%}
%%
[0-9]+      { yylval = atoi(yytext); return NUMBER; }
[+\-*/^n]   { return yytext[0]; }
\n          { return '\n'; }
[ \t]       { /* Ignore whitespaces */ }
.           { printf("Invalid character: %s\n", yytext); exit(1); }
%%
int yywrap() {
    return 1;
}
```

OUTPUT:



```
student@oslab-02:~/220905128/lab10/q4$ ./q4
Enter a postfix expression:
5+3
Invalid expression
student@oslab-02:~/220905128/lab10/q4$ ./q4
Enter a postfix expression:
5 3 + 2 *
Result = 16
5 + 2
Invalid expression
```