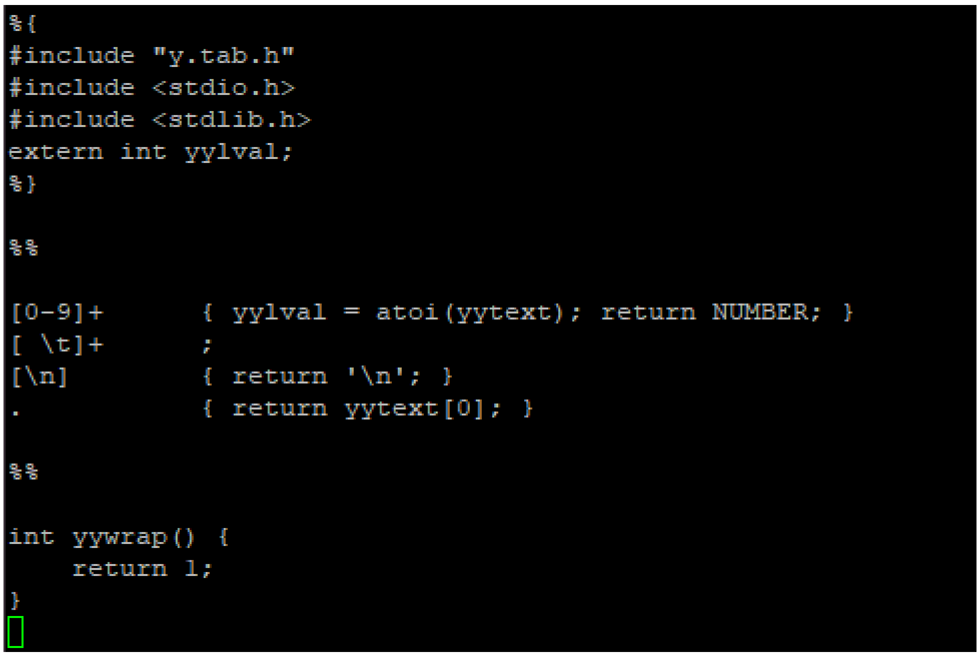
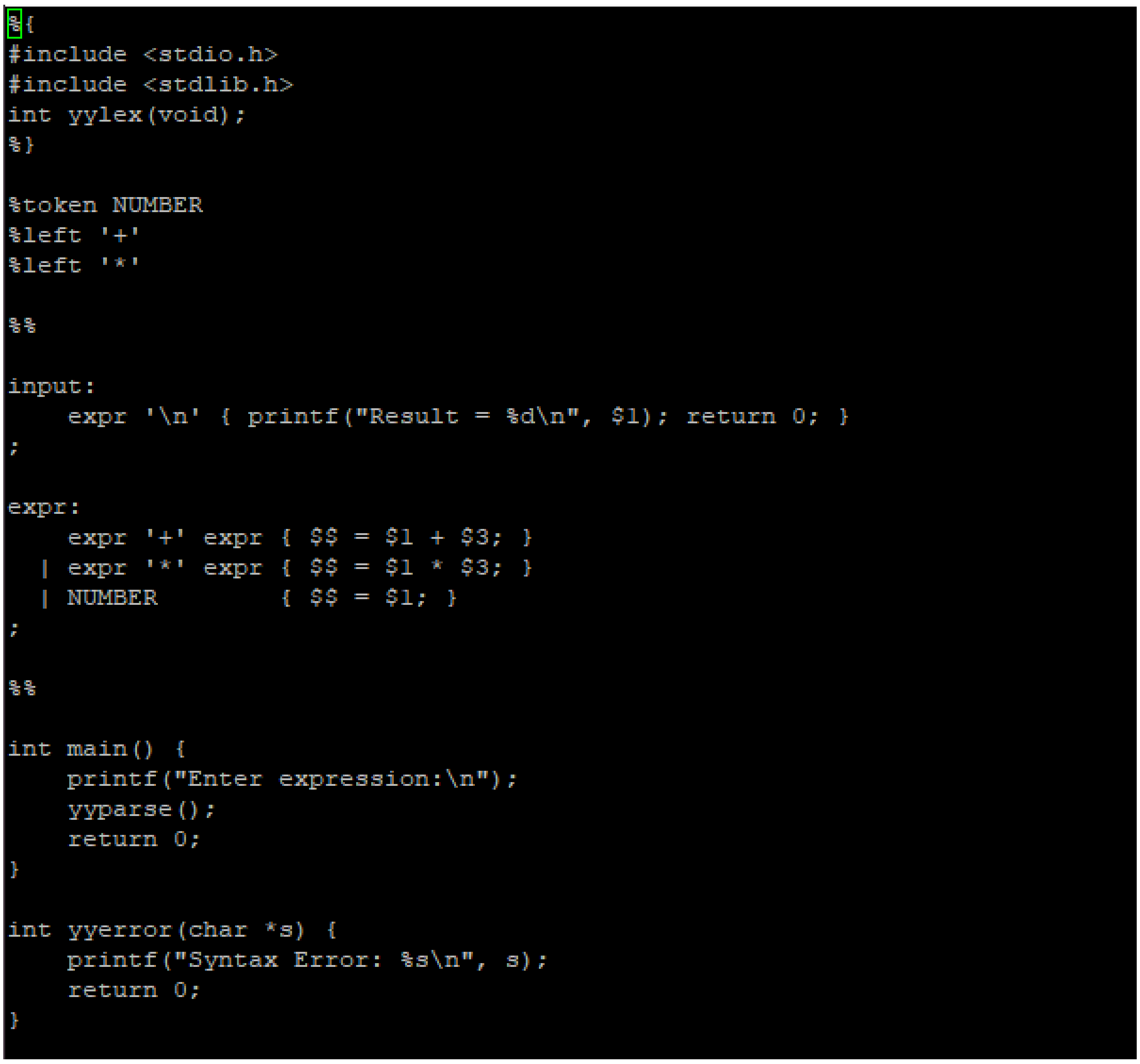
expr.l



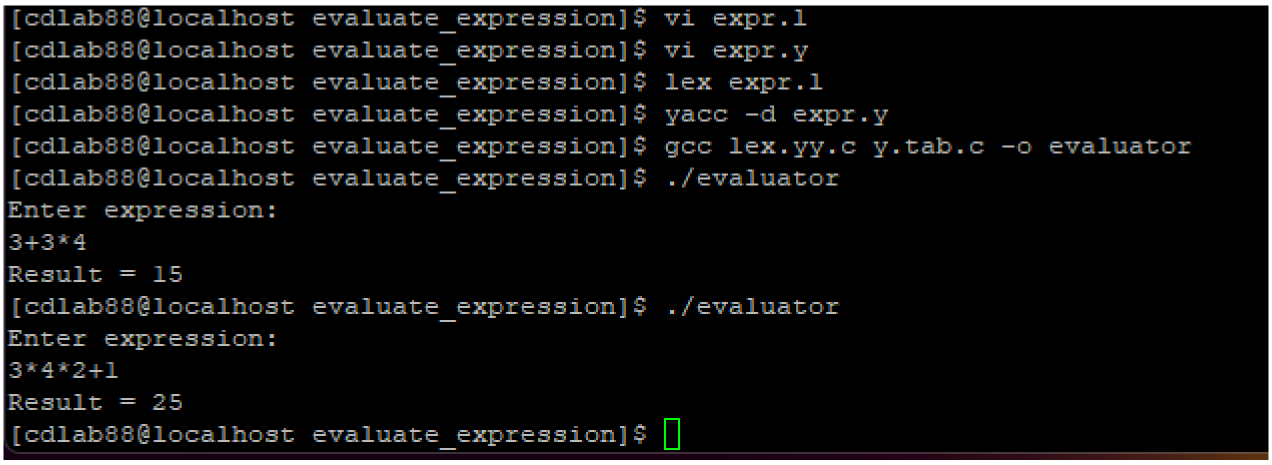
```
%{
#include "y.tab.h"
#include <stdio.h>
#include <stdlib.h>
extern int yylval;
%}

%%

[0-9]+        { yylval = atoi(yytext); return NUMBER; }
[ \t]+        ;
[\n]          { return '\n'; }
.             { return yytext[0]; }

%%

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

expr.y



```
%{
#include <stdio.h>
#include <stdlib.h>
int yylex(void);
%}

%token NUMBER
%left '+'
%left '*'

%%

input:
    expr '\n' { printf("Result = %d\n", $1); return 0; }
;

expr:
    expr '+' expr { $$ = $1 + $3; }
  | expr '*' expr { $$ = $1 * $3; }
  | NUMBER        { $$ = $1; }
;

%%

int main() {
    printf("Enter expression:\n");
    yyparse();
    return 0;
}

int yyerror(char *s) {
    printf("Syntax Error: %s\n", s);
    return 0;
}
```

OUTPUT



```
[cdlab88@localhost evaluate_expression]$ vi expr.l
[cdlab88@localhost evaluate_expression]$ vi expr.y
[cdlab88@localhost evaluate_expression]$ lex expr.l
[cdlab88@localhost evaluate_expression]$ yacc -d expr.y
[cdlab88@localhost evaluate_expression]$ gcc lex.yy.c y.tab.c -o evaluator
[cdlab88@localhost evaluate_expression]$ ./evaluator
Enter expression:
3+3*4
Result = 15
[cdlab88@localhost evaluate_expression]$ ./evaluator
Enter expression:
3*4*2+1
Result = 25
[cdlab88@localhost evaluate_expression]$
```