



SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

Symbiosis International (Deemed University)

(Established under section 3 of the UGC Act, 1956)

Re-accredited by NAAC with 'A' grade (3.58/4) | Awarded Category – I by UGC

Founder: Prof. Dr. S. B. Mujumdar, M. Sc., Ph. D. (Awarded Padma Bhushan and Padma Shri by President of India)

Assignment No. 08

Subject:	Compiler Construction Lab
Name of Student	Onkar Mendhapurkar
PRN No.	22070122135
Branch	CSE B2, Batch (2022-26)
Academic Year & Semester	2022-26
Date of Performance	23/09/2025
Title of Assignment:	Desk calculator with error recovery.
Practice Questions	<ol style="list-style-type: none">1. YACC program for desk calculator with error recovery.2. YACC program for desk calculators to evaluate arithmetic expressions involving parenthesis. <p>PostLab Question</p> <ol style="list-style-type: none">3. LEX program for desk calculator
Source Code	<pre>1. deskcalk.l %{ #include "deskcalk.tab.h" #include <stdlib.h> %} %% [0-9]+(\\.[0-9]+)? { yylval.fval = atof(yytext); return NUMBER; } [+\\-*/()] { return yytext[0]; } \\n { return '\\n'; } [\\t] { /* ignore spaces */ } . { /* ignore other characters */ }</pre>

```
%%
```

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

deskcalk.y

```
%{
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int yylex();
```

```
void yyerror(const char *s);
```

```
%}
```

```
%union {
```

```
    float fval;
```

```
}
```

```
%token <fval> NUMBER
```

```
%type <fval> expr
```

```
%left '+' '-'
```

```
%left '*' '/'
```

```
%right UMINUS
```

```
%%
```

```
input:
```

```
    /* empty */
```

```
    | input line
```

```
    ;
```

```
line:
```

```
    '\n'
```

```
    | expr '\n'      { printf("Result = %.2f\n", $1); }
```

```
    | error '\n'     { yyerror("Invalid expression"); yyerrok; }
```

```
    ;
```

```
expr:
```

```
    NUMBER           { $$ = $1; }
```

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

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

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

```

| expr '/' expr      {
                        if ($3 == 0) { yyerror("Division by zero"); $$ = 0; }
                        else { $$ = $1 / $3; }
                      }
| '(' expr ')'        { $$ = $2; }
| '-' expr %prec UMINUS { $$ = -$2; }
;

%%

void yyerror(const char *s) {
    fprintf(stderr, "Error: %s\n", s);
}

int main() {
    printf("Desk Calculator with error recovery\n");
    printf("Enter expressions (press Ctrl+D to exit):\n");
    yyparse();
    return 0;
}

```

2.

deskcalk_paren.l

```

%{
#include "deskcalk_paren.tab.h"
#include <stdlib.h>
%}

%%

[0-9]+(\.[0-9]+)? { yylval.fval = atof(yytext); return NUMBER; }
[+|-*|/]          { return yytext[0]; }
\n                { return '\n'; }
[ \t]             { /* ignore spaces and tabs */ }
.                 { /* ignore invalid characters */ }

%%

int yywrap() { return 1; }

```

deskcalk_paren.y

```

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

int yylex();
void yyerror(const char *s);
%}

%union {
    float fval;
}

%token <fval> NUMBER
%type <fval> expr

%left '+' '-'
%left '*' '/'
%right UMINUS

%%

input:
    /* empty */
    | input line
    ;

line:
    '\n'
    | expr '\n'    { printf("Result = %.2f\n", $1); }
    | error '\n'   { yyerror("Invalid expression"); yyerrok; }
    ;

expr:
    NUMBER          { $$ = $1; }
    | expr '+' expr  { $$ = $1 + $3; }
    | expr '-' expr  { $$ = $1 - $3; }
    | expr '*' expr  { $$ = $1 * $3; }
    | expr '/' expr  {
                        if ($3 == 0) { yyerror("Division by zero"); $$ = 0; }
                        else { $$ = $1 / $3; }
                    }
    | '(' expr ')'    { $$ = $2; }
    | '-' expr %prec UMINUS { $$ = -$2; }
    ;

```

```
%%
```

```
void yyerror(const char *s) {  
    fprintf(stderr, "Error: %s\n", s);  
}
```

```
int main() {  
    printf("Desk Calculator with Parentheses\n");  
    printf("Enter expressions (press Ctrl+D to exit):\n");  
    yyparse();  
    return 0;  
}
```

3.

deskcalk.l

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

```
%%
```

```
[0-9]+(\.[0-9]+)? { printf("NUMBER: %s\n", yytext); }  
[+|-|*|/] { printf("OPERATOR: %s\n", yytext); }  
\n { printf("NEWLINE\n"); }  
[ \t] { /* ignore spaces and tabs */ }  
. { printf("UNKNOWN: %s\n", yytext); }
```

```
%%
```

```
int main() {  
    printf("Enter expressions (Ctrl+D to exit):\n");  
    yylex();  
    return 0;  
}
```

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

Output Screenshot

1.

```
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ nano deskcalc.l
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ nano deskcalc.y
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ bison -d deskcalc.y
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ flex deskcalc.l
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ ls
deskcalc.l  deskcalc.tab.c  deskcalc.tab.h  deskcalc.y  lex.yy.c
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ gcc lex.yy.c deskcalc.tab.c -o deskcalc -lm
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ ./deskcalc
Desk Calculator with error recovery
Enter expressions (press Ctrl+D to exit):
3+5*2
Result = 13.00
(4+5)/0
Error: Division by zero
Result = 0.00
2++3
Error: syntax error
Error: Invalid expression
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ |
```

2.

```
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ nano deskcalc_paren.l
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ nano deskcalc_paren.y
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ bison -d deskcalc_paren.y
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ flex deskcalc_paren.l
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ gcc lex.yy.c deskcalc_paren.tab.c -o deskcalc_paren -lm
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ ls
deskcalc  deskcalc.tab.c  deskcalc.y  deskcalc_paren.l  deskcalc_paren.tab.h  lex.yy.c
deskcalc.l  deskcalc.tab.h  deskcalc_paren  deskcalc_paren.tab.c  deskcalc_paren.y
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ ./deskcalc_paren
Desk Calculator with Parentheses
Enter expressions (press Ctrl+D to exit):
3+5*2
Result = 13.00
(4+5)*2
Result = 18.00
7/(2-2)
Error: Division by zero
Result = 0.00
2++3
Error: syntax error
Error: Invalid expression
^C
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ |
```

3. PostLab Experiment

```
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ nano deskcalc.l
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ flex deskcalc.l
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ gcc lex.yy.c -o deskcalc -lm
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ ls
deskcalc  deskcalc.tab.c  deskcalc.y  deskcalc_paren.l  deskcalc_paren.tab.h  lex.yy.c
deskcalc.l  deskcalc.tab.h  deskcalc_paren  deskcalc_paren.tab.c  deskcalc_paren.y
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ ./deskcalc
Enter expressions (Ctrl+D to exit):
3+5*2
NUMBER: 3
OPERATOR: +
NUMBER: 5
OPERATOR: *
NUMBER: 2
NEWLINE
(4+6)*2
OPERATOR: (
NUMBER: 4
OPERATOR: +
NUMBER: 6
OPERATOR: )
OPERATOR: *
NUMBER: 2
NEWLINE
^C
battlemachine@DESKTOP-FU1975B:~/CCL/Exp8$ |
```

Conclusion

These experiments demonstrate how LEX and YACC can be combined to parse and evaluate arithmetic expressions, including handling decimals, parentheses, unary operators, and errors. They highlight the effectiveness of syntax analysis, operator precedence, and error recovery in building robust expression evaluators.