



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. 07

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	09/09/2025
Title of Assignment:	Postfix Expression Evaluation.
Practice Questions	<div>1. YACC program for Postfix Expression Evaluation.</div> <div>2. YACC program for Conversion of Infix to Postfix expression.</div> <div>PostLab Question</div> <div>3.YACC program for evaluating postfix expressions containing decimal numbers.</div>
Source Code	<div>1.</div> <div>postfix.l</div> <div>%{ #include "postfix.tab.h" #include <stdlib.h> %} %% [0-9] { yyval = atoi(yytext); return NUMBER; } [+\\-*/] { return yytext[0]; } \\n { return '\\n'; } . { /* ignore other characters */ } %%</div>

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

postfix.y

```
%{
```

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

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

```
int yylex();
```

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

```
%}
```

```
%token NUMBER
```

```
%%
```

```
input:
```

```
    /* empty */
```

```
    | input line
```

```
    ;
```

```
line:
```

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

```
    ;
```

```
expr:
```

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

```
    | expr expr '+'  { $$ = $1 + $2; }
```

```
    | expr expr '-'  { $$ = $1 - $2; }
```

```
    | expr expr '*'  { $$ = $1 * $2; }
```

```
    | expr expr '/'  { $$ = $1 / $2; }
```

```
    ;
```

```
%%
```

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

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

```
}
```

```
int main() {
```

```
    printf("Enter postfix expression:\n");
```

```
    yyparse();
```

```
    return 0;
```

```
}
```

2. **infix.l**

```
%{  
#include "infix.tab.h"  
%}  
  
%%  
[a-zA-Z] { yylval = *yytext; return ID; }  
[+\\-*/()] { return yytext[0]; }  
\\n      { return '\\n'; }  
[ \\t]    { /* ignore spaces */ }  
\\.       { /* ignore other characters */ }  
%%
```

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

infix.y

```
%{  
#include <stdio.h>  
#include <stdlib.h>  
  
int yylex();  
void yyerror(const char *s);  
void print(char c);  
%}
```

```
%token ID
```

```
%%  
input:  
    /* empty */  
    | input line  
    ;
```

```
line:  
    expr '\\n' { printf("\\n"); }  
    ;
```

```
expr:  
    expr '+' expr { print('+'); }  
    | expr '-' expr { print('-'); }
```

```

| expr '*' expr { print('*'); }
| expr '/' expr { print('/'); }
| '(' expr ')' { /* brackets handled */ }
| ID           { print($1); }
;
%%

```

```

void print(char c) {
    printf("%c", c);
}

```

```

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

```

```

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

```

3. postfix_decimal.l

```

%{
#include "postfix_decimal.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; }

```

postfix_decimal.y

```

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

```

```

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

%union {
    float fval;
}

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

%%

input:
    /* empty */
    | input line
    ;

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

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

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

int main() {
    printf("Enter postfix expression (with decimals):\n");

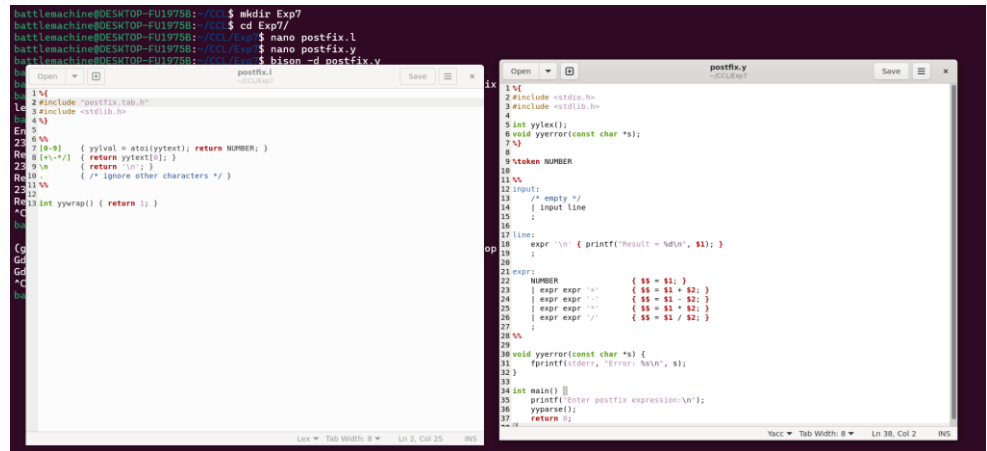
```

```
yyparse();  
return 0;  
}
```

Output Screenshot

1.

```
battlemachine@DESKTOP-FU1975B:~/CCL$ mkdir Exp7  
battlemachine@DESKTOP-FU1975B:~/CCL$ cd Exp7/  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ nano postfix.l  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ nano postfix.y  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ bison -d postfix.y  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ flex postfix.l  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ gcc lex.yy.c postfix.tab.c -o postfix  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ ls  
lex.yy.c postfix postfix.l postfix.tab.c postfix.tab.h postfix.y  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ ./postfix  
Enter postfix expression:  
23*54**+9-  
Result = 17  
23*54**+9-  
Result = 17  
234**+  
Result = 14  
^C
```



2.

```
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ nano infix.l  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ nano infix.y  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ bison -d infix.y  
infix.y: warning: 16 shift/reduce conflicts [-Wconflicts-sr]  
infix.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ flex infix.l  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ gcc lex.yy.c infix.tab.c -o infix  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ ls  
infix infix.l infix.tab.c infix.tab.h infix.y lex.yy.c postfix postfix.l postfix.tab.c postfix.tab.h postfix.y  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$ ./infix  
Enter infix expression:  
a+b*c-d  
abcd-+*  
a+b  
ab+  
(a+b)*c  
ab+c*  
a+b*c-d  
abcd-+*  
^C  
battlemachine@DESKTOP-FU1975B:~/CCL/Exp7$
```

