



॥ वसुधैव कुटुम्बकम् ॥

SYMBIOSIS INSTITUTE OF TECHNOLOGY, NAGPUR

Constituent of Symbiosis International (Deemed University), Pune

(Established under Section 3 of the UGC Act of 1956 wide notification number F-9-12/2001-U-3 of Government of India)

Re-Accredited by NAAC with 'A++' Grade



Compiler Construction Lab

1 - 10 Practical

Submitted To:

Dr. Snehlata Wankhade

Submitted By:

Suhani Gahukar

22070521084

VIIth sem , C

PRACTICAL 1

AIM - Theory assignment for writing details about LEX and YACC compilation.

```
%{
#include<stdio.h>
%}
%%
zero|ZERO|Zero printf("0");
one|ONE|One printf("1");
two|TWO|Two printf("2");
three|THREE|Three printf("3");
four|Four|Four printf("4");
five|FIVE|Five printf("5");
six|SIX|Six printf("6");
seven|SEVEN|Seven printf("7");
eight|EIGHT|Eight printf("8");
nine|NINE|Nine printf("9");
%%
int main()
{
yylex();
return 0;
}
```

Open ▾



file084.l

~/

```
1 %{
2 #include<stdio.h>
3 %}
4 %%
5 zero|ZERO|Zero printf("0");
6 one|ONE|One printf("1");
7 two|TWO|Two printf("2");
8 three|THREE|Three printf("3");
9 four|Four|Four printf("4");
10 five|FIVE|Five printf("5");
11 six|SIX|Six printf("6");
12 seven|SEVEN|Seven printf("7");
13 eight|EIGHT|Eight printf("8");
14 nine|NINE|Nine printf("9");
15 %%
16 int main()
17 {
18     yylex();
19     return 0;
20 }
```

```
^C
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ gedit file084.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ lex file084.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ cc lex.yy.c -ll
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ ./a.out
one
1
two
2
TWO
2
three
3
SIX
6
nine
9
ZERO
0
four
4
Five
5
seven
7
EIGHT
8
^C
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$
```

PRACTICAL 2

AIM - Count the number of comments, keywords, identifiers, words, lines and spaces from input file.

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

int lc = 0, sc = 0, wc = 0, ch = 0;
int kw = 0, id = 0, cm = 0;

char *keywords[] = {
    "int", "float", "return", "if", "else", "for", "while", "char", "double", "void", NULL
};

int is_keyword(char *word) {
    for (int i = 0; keywords[i]; i++) {
        if (strcmp(word, keywords[i]) == 0)
            return 1;
    }
    return 0;
}
}%

%%

/*".*          { cm++; ch+=yyleng; } // Single-line comment
/*"([^\]|*+[^/] )*\*+"/*"  { cm++; ch+=yyleng; } // Multi-line comment
[a-zA-Z_][a-zA-Z0-9_]*      {
    if (is_keyword(yytext))
        kw++;
    else
        id++;
    wc++;
    ch += yyleng;
}
[ \t]+          { sc += yyleng; ch += yyleng; }
\n              { lc++; ch++; }
.               { ch++; } // Any other character
}%

int yywrap() { return 1; }

int main() {
    printf("Enter the input:\n");
    yylex();
    printf("Lines    : %d\n", lc);
```

```
printf("Spaces   : %d\n", sc);
printf("Characters: %d\n", ch);
printf("Words    : %d\n", wc);
printf("Identifiers: %d\n", id);
printf("Keywords  : %d\n", kw);
printf("Comments  : %d\n", cm);
return 0;
}
```

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

int lc = 0, sc = 0, wc = 0, ch = 0;
int kw = 0, id = 0, cm = 0;

char *keywords[] = {
    "int", "float", "return", "if", "else", "for", "while", "char", "double", "void", NULL
};

int is_keyword(char *word) {
    for (int i = 0; keywords[i]; i++) {
        if (strcmp(word, keywords[i]) == 0)
            return 1;
    }
    return 0;
}

%%
"/*".* { cm++; // Single-line comment
        ch += yyleng;
    }
"/*"([^\*]|\*+[/])*\*+/" { cm++; // Multi-line comment
                            ch += yyleng;
    }
[a-zA-Z][a-zA-Z0-9_]* { if (is_keyword(yytext))
                        kw++; // Keyword count
                        else
                            id++; // Identifier count
                        wc++; // Word count
                        ch += yyleng; // Character count
    }
[ \t]+ { sc += yyleng; // Space count
        ch += yyleng;
    }
\n { lc++; // Line count
    ch++; // Character count (newline)
    }
. { ch++; // Any other character
    }
%%

int yywrap() {
    return 1;
}

int main() {
    printf("Enter the input:\n");
    yylex();
    printf("Lines : %d\n", lc);
    printf("Spaces : %d\n", sc);
    printf("Characters: %d\n", ch);
    printf("Words : %d\n", wc);
    printf("Identifiers: %d\n", id);
    printf("Keywords : %d\n", kw);
    printf("Comments : %d\n", cm);
    return 0;
}
```

```
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ lex file_084.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ cc lex.yy.c -ll
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ ./a.out
Enter the input:
I am Suhani Gahukar.
/*SIT nagpur
*/
//4th year
if it rained today
else I need to go to cllg
NULL
Lines      : 6
Spaces    : 13
Characters: 99
Words     : 16
Identifiers: 14
Keywords  : 2
Comments  : 2
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$
```

PRACTICAL 3

AIM - Count the number of words starting with 'A'.

```
%{
#include <stdio.h>
int count = 0;
}%
%%
[Aa][a-zA-Z]* { count++; }
.|\\n ;
%%

int main() {
yylex();
printf("Number of words starting with 'A' or 'a' : %d\\n", count);
return 0;
}

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

```
1 %{
2 #include <stdio.h>
3 int count = 0;
4 %}
5
6 %%
7 [Aa][a-zA-Z]*      { count++; }
8 .|\\n              ;
9 %%
10
11 int main() {
12     yylex();
13     printf("Number of words starting with 'A' or 'a' : %d\\n", count);
14     return 0;
15 }
16
17 int yywrap() {
18     return 1;
19 }
```

```
suhami@LAPTOP-OTTF73G6:~/CC$ flex file_03_084.l
suhami@LAPTOP-OTTF73G6:~/CC$ gcc lex.yy.c -o countA
suhami@LAPTOP-OTTF73G6:~/CC$ ./countA
Apple and Ants are amazing animals.
Number of words starting with 'A' or 'a' : 6
suhami@LAPTOP-OTTF73G6:~/CC$ |
```

PRACTICAL 4

AIM - Conversion of lowercase to uppercase and vice versa.

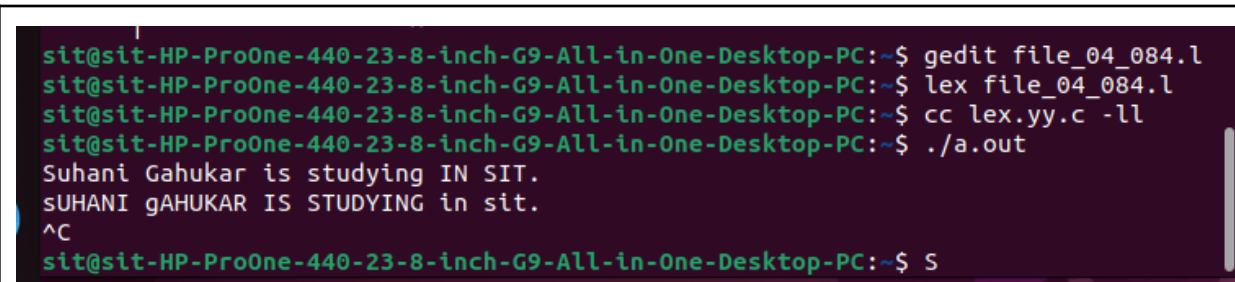
```
%{
#include<stdio.h>
%}
%%
[a-z] { printf("%c", yytext[0] - 32); } // Convert lowercase to uppercase
[A-Z] { printf("%c", yytext[0] + 32); } // Convert uppercase to lowercase
.\n { printf("%s", yytext); }

%%

int main() {
yylex();
return 0;
}
int yywrap() {
return 1;
}
```



```
1 %{
2 #include<stdio.h>
3 %}
4 %%
5 [a-z] { printf("%c", yytext[0] - 32); } // Convert lowercase to uppercase
6 [A-Z] { printf("%c", yytext[0] + 32); } // Convert uppercase to lowercase
7 .\n { printf("%s", yytext); }
8
9 %%
10
11 int main() {
12 yylex();
13 return 0;
14 }
15 int yywrap() {
16 return 1;
17 }
```



```
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ gedit file_04_084.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ lex file_04_084.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ cc lex.yy.c -ll
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ ./a.out
Suhani Gahukar is studying IN SIT.
SUHANI gAHUKAR IS STUDYING in sit.
^C
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ S
```


PRACTICAL 5

AIM - Conversion of decimal to hexadecimal number in a file.

```
%{
#include <stdio.h>
#include <string.h>
void decimal_to_hex(int num) {
    char hex[100];
    int i = 0, remainder;
    if(num == 0) {
        printf("0x0\n");
        return;
    }
    while(num != 0) {
        remainder = num % 16;
        if(remainder < 10) hex[i++] = remainder + '0';
        else hex[i++] = remainder - 10 + 'A';
        num /= 16;
    }
    printf("0x");
    for(int j = i - 1; j >= 0; j--) printf("%c", hex[j]);
}
int string_to_int(char *str) {
    int result = 0;
    for(int i=0; str[i] != '\0'; i++) result = result * 10 + (str[i] - '0');
    return result;
}
}%

%%
[0-9]+ { decimal_to_hex(string_to_int(yytext)); }
.\n { ECHO; }
%%

int main() {
    printf("Enter decimal numbers (Ctrl+D to end):\n");
    yylex();
    return 0;
}
int yywrap() {
    return 1;
}
```

```
Open  prms_084.l
1 %{
2 #include <stdio.h>
3 %}
4 %%
5 [0-9]+ {
6     int decimal_value = 0;
7     char hex_string[100]; // Buffer to store the hexadecimal digits
8     int i = 0;
9
10    // Manual conversion from string to integer
11    for (int k = 0; yytext[k] != '\0'; k++) {
12        decimal_value = decimal_value * 10 + (yytext[k] - '0');
13    }
14
15    // Convert and print hexadecimal
16    if (decimal_value == 0) {
17        printf("0x0\n");
18    } else {
19        while (decimal_value > 0) {
20            int remainder = decimal_value % 16;
21            if (remainder < 10) {
22                hex_string[i] = remainder + '0';
23            } else {
24                hex_string[i] = remainder - 10 + 'A';
25            }
26            decimal_value = decimal_value / 16;
27            i++;
28        }
29        printf("0x");
30        for (int j = i - 1; j >= 0; j--) {
31            printf("%c", hex_string[j]);
32        }
33        printf("\n");
34    }
35 }
36
37
38 .|\n {
39     putchar(yytext[0]);
40 }
41
42 %%
43
44 int main() {
45     yylex();
46     return 0;
47 }
48 int yywrap() {
49     return 1;
50 }
51
```

```
prms_084.l:3: unrecognized rule
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ gedit prms_084.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ lex prms_084.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ cc lex.yy.c -ll
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ ./a.out
123
0x7B

255
0xFF

789
0x315

12
0xC

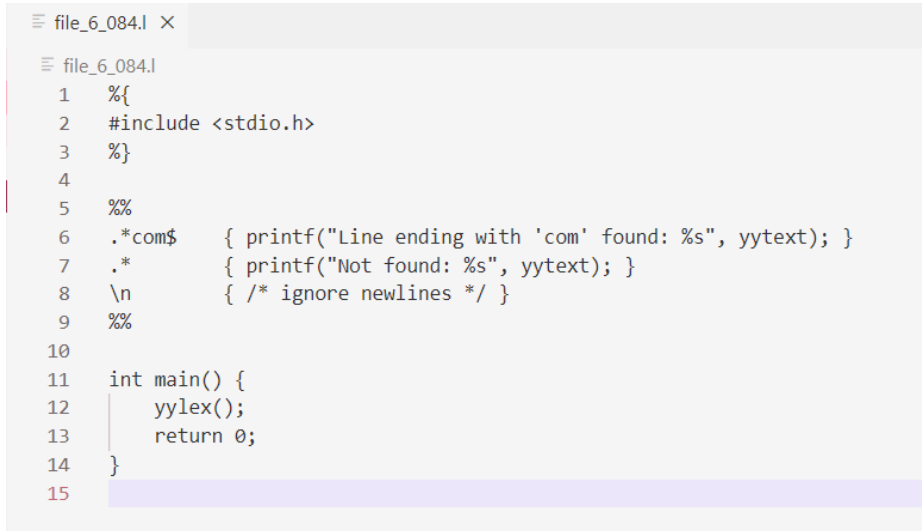
16
0x10

^C
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$
```

PRACTICAL 6

AIM - Test lines ending with "com".

```
%{  
#include <stdio.h>  
%}  
  
%%  
.*com$  { printf("Line ending with 'com' found: %s", yytext); }  
.*      { printf("Not found: %s", yytext); }  
\n      { /* ignore newlines */ }  
%%  
  
int main() {  
    yylex();  
    return 0;  
}
```



```
file_6_084.l  
1  %{  
2  #include <stdio.h>  
3  %}  
4  
5  %%  
6  .*com$  { printf("Line ending with 'com' found: %s", yytext); }  
7  .*      { printf("Not found: %s", yytext); }  
8  \n      { /* ignore newlines */ }  
9  %%  
10  
11 int main() {  
12     yylex();  
13     return 0;  
14 }  
15
```

```
suhami@LAPTOP-OTTF73G6:~/CC$ flex file_6_084.l  
suhami@LAPTOP-OTTF73G6:~/CC$ cc lex.yy.c -ll  
suhami@LAPTOP-OTTF73G6:~/CC$ ./a.out  
website.in  
Not found: website.in  
  
hello.com  
Line ending with 'com' found: hello.com  
  
exmaple.co  
Not found: exmaple.co  
  
exmaple.com
```

PRACTICAL 7

AIM - Postfix Expression Evaluation.

file.y

```
%{
#include <stdio.h>
#include <stdlib.h>
#define YYSTYPE int

int stack[100];
int top = -1;

#define PUSH(v) (stack[++top] = (v))
#define POP() (stack[top--])

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

%token NUMBER

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

line:
    elements 'n'
    {
        if (top >= 0) {
            printf("Result = %d\n", stack[top]);
            /* reset stack for next line */
            top = -1;
        } else {
            printf("No result (empty expression)\n");
        }
    }
    ;

elements:
    /* zero or more elements (numbers or operators) */
    | elements element
    ;

element:
```

```

NUMBER      { PUSH($1); }
| '+'      {
            if (top < 1) { yyerror("not enough operands for +"); }
            else { int b = POP(); int a = POP(); PUSH(a + b); }
            }
| '-'      {
            if (top < 1) { yyerror("not enough operands for -"); }
            else { int b = POP(); int a = POP(); PUSH(a - b); }
            }
| '*'      {
            if (top < 1) { yyerror("not enough operands for *"); }
            else { int b = POP(); int a = POP(); PUSH(a * b); }
            }
| '/'      {
            if (top < 1) { yyerror("not enough operands for /"); }
            else { int b = POP(); int a = POP();
                  if (b == 0) { yyerror("division by zero"); }
                  else PUSH(a / b);
            }
            }
;
%%
void yyerror(const char *s) {
    /* simple error printer - doesn't exit parser to let other lines be read */
    fprintf(stderr, "Error: %s\n", s);
    /* reset stack to avoid cascading errors */
    top = -1;
}

int main(void) {
    printf("Postfix evaluator (enter one postfix expression per line):\n");
    yyparse();
    return 0;
}

```

postfix_084.l

postfix_084.y ×

postfix_084.y

```
1  %{
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define YYSTYPE int
5
6  int stack[100];
7  int top = -1;
8
9  #define PUSH(v) (stack[++top] = (v))
10 #define POP() (stack[top--])
11
12 void yyerror(const char *s);
13 int yylex(void);
14 %}
15
16 %token NUMBER
17
18 %%
19 input:
20 | /* empty */
21 | input line
22 ;
23
24 line:
25 | elements '\n' {
26 | { if (top >= 0) {
27 |   printf("Result = %d\n", stack[top]);
28 |   /* reset stack for next line */
29 |   top = -1;
30 | } else {
31 |   printf("No result (empty expression)\n");
32 | }
33 | }
34 ;
35
36 elements:
37 | /* zero or more elements (numbers or operators) */
38 | elements element
39 ;
40
41 element:
42 | NUMBER { PUSH($1); }
43 | '+' {
44 |   if (top < 1) { yyerror("not enough operands for +"); }
45 |   else { int b = POP(); int a = POP(); PUSH(a + b); }
```

```

postfix_084.l  postfix_084.y ×
postfix_084.y
41  element:
43      | '+'          {
46      |             }
47      | '-'          {
48      |             if (top < 1) { yyerror("not enough operands for -"); }
49      |             else { int b = POP(); int a = POP(); PUSH(a - b); }
50      |             }
51      | '*'          {
52      |             if (top < 1) { yyerror("not enough operands for *"); }
53      |             else { int b = POP(); int a = POP(); PUSH(a * b); }
54      |             }
55      | '/'          {
56      |             if (top < 1) { yyerror("not enough operands for /"); }
57      |             else { int b = POP(); int a = POP();
58      |                 if (b == 0) { yyerror("division by zero"); }
59      |                 else PUSH(a / b);
60      |             }
61      |             }
62      ;
63  %%
64  void yyerror(const char *s) {
65      /* simple error printer - doesn't exit parser to let other lines be read */
66      fprintf(stderr, "Error: %s\n", s);
67      /* reset stack to avoid cascading errors */
68      top = -1;
69  }
70
71  int main(void) {
72      printf("Postfix evaluator (enter one postfix expression per line):\n");
73      yyparse();
74      return 0;
75  }

```

file.l

```

%{
#include "y.tab.h"
extern int yylval;
%}
%%
[ \t]+      ;      /* skip whitespace */
[0-9]+      {
/* convert yytext (string of digits) to integer */
int i = 0;
int val = 0;
while (yytext[i]) {
val = val * 10 + (yytext[i] - '0');
i++;
}
yylval = val;
return NUMBER;
}
\n          { return '\n'; }
"+"         { return '+'; }
"-"         { return '-'; }
"*"         { return '*'; }

```

```

"/"      { return '/'; }
.        { fprintf(stderr, "Unknown character: %s\n", yytext); }
%%

```

```

postfix_084.l
1  %{
2  #include "y.tab.h"
3  extern int yylval;
4  %}
5  %%
6  [ \t]+      ;          /* skip whitespace */
7  [0-9]+      {
8                  /* convert yytext (string of digits) to integer */
9                  int i = 0;
10                 int val = 0;
11                 while (yytext[i]) {
12                     val = val * 10 + (yytext[i] - '0');
13                     i++;
14                 }
15                 yylval = val;
16                 return NUMBER;
17             }
18  \n          { return '\n'; }
19  "+"         { return '+'; }
20  "-"         { return '-'; }
21  "*"         { return '*'; }
22  "/"         { return '/'; }
23  .           { fprintf(stderr, "Unknown character: %s\n", yytext); }
24  %%

```

```

suhami@LAPTOP-OTTF73G6:~/CC$ yacc -d postfix_084.y
suhami@LAPTOP-OTTF73G6:~/CC$ lex postfix_084.l
suhami@LAPTOP-OTTF73G6:~/CC$ gcc y.tab.c lex.yy.c -o calssd -lfl
suhami@LAPTOP-OTTF73G6:~/CC$ ./calssd
Postfix evaluator (enter one postfix expression per line):
3 4 +
Result = 7
10 5 -
Result = 5
2 3 4 * +
Result = 14
20 4 /
Result = 5
5 1 2 + 4 * + 3 -
Result = 14
^C

```


PRACTICAL 8

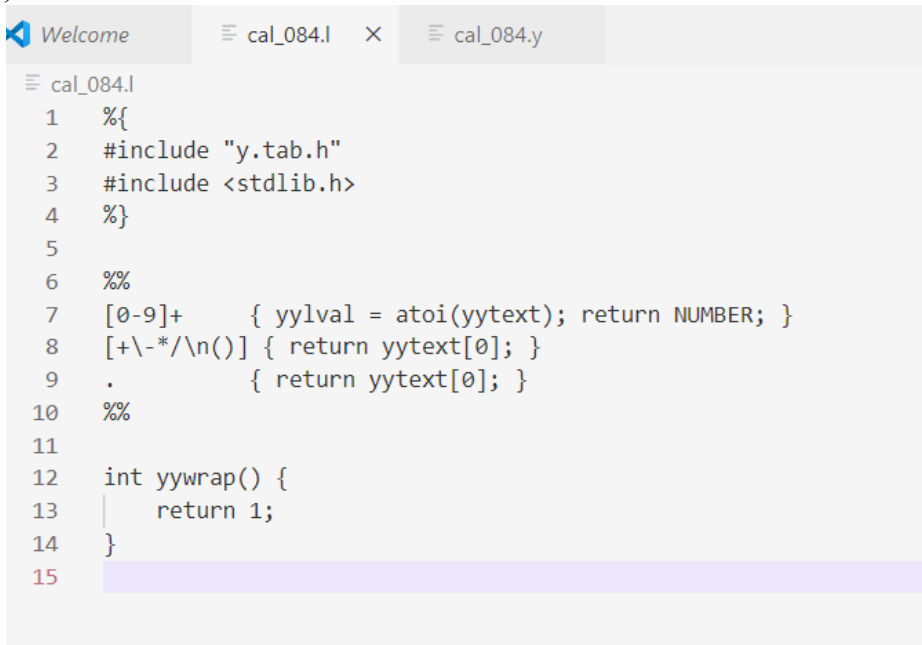
AIM - Desk calculator with error recovery.

cal.l

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

%%
[0-9]+ { yylval = atoi(yytext); return NUMBER; }
[+\\-*/\\n()] { return yytext[0]; }
.      { return yytext[0]; }
%%

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



cal.y

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

%token NUMBER
%left '+' '-'
%left '*' '/'

%%
```

```

input : /* empty */
    | input line
    ;
line : expr '\n' { printf("Result = %d\n", $1); }
    | error '\n' { printf("Syntax Error! Please re-enter.\n"); yyerror; }
    ;
expr : expr '+' expr { $$ = $1 + $3; }
    | expr '-' expr { $$ = $1 - $3; }
    | expr '*' expr { $$ = $1 * $3; }
    | expr '/' expr { if ($3 == 0) { printf("Error: Division by zero!\n"); $$ = 0; } else $$ = $1 / $3; }
    | '(' expr ')' { $$ = $2; }
    | NUMBER
    ;
%%

```

```

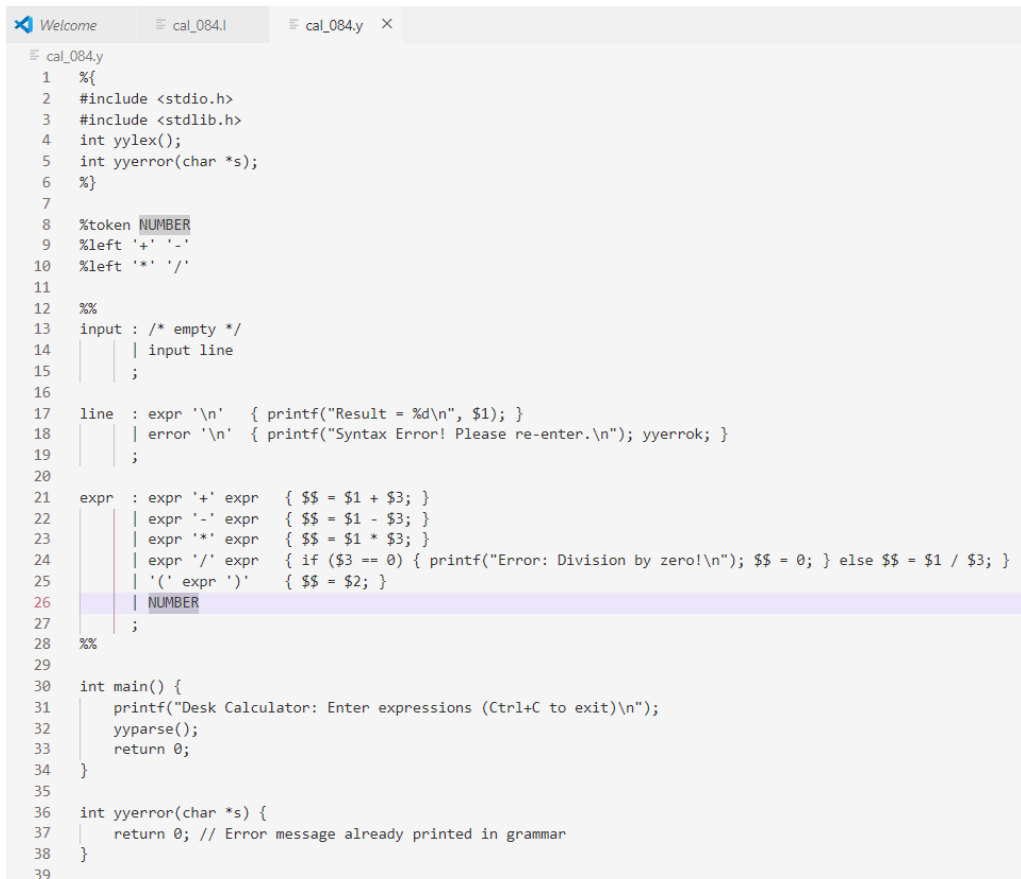
int main() {
    printf("Desk Calculator: Enter expressions (Ctrl+C to exit)\n");
    yyparse();
    return 0;
}

```

```

int yyerror(char *s) {
    return 0; // Error message already printed in grammar
}

```



```

Welcome  cal_084.l  cal_084.y  X
cal_084.y
1  %{
2  #include <stdio.h>
3  #include <stdlib.h>
4  int yylex();
5  int yyerror(char *s);
6  %{
7
8  %token NUMBER
9  %left '+' '-'
10 %left '*' '/'
11
12 %%
13 input : /* empty */
14     | input line
15     ;
16
17 line : expr '\n' { printf("Result = %d\n", $1); }
18     | error '\n' { printf("Syntax Error! Please re-enter.\n"); yyerror; }
19     ;
20
21 expr : expr '+' expr { $$ = $1 + $3; }
22     | expr '-' expr { $$ = $1 - $3; }
23     | expr '*' expr { $$ = $1 * $3; }
24     | expr '/' expr { if ($3 == 0) { printf("Error: Division by zero!\n"); $$ = 0; } else $$ = $1 / $3; }
25     | '(' expr ')' { $$ = $2; }
26     | NUMBER
27     ;
28 %%
29
30 int main() {
31     printf("Desk Calculator: Enter expressions (Ctrl+C to exit)\n");
32     yyparse();
33     return 0;
34 }
35
36 int yyerror(char *s) {
37     return 0; // Error message already printed in grammar
38 }
39

```

```
suhami@LAPTOP-OTTF73G6:~/CC$ yacc -d cal_084.y
suhami@LAPTOP-OTTF73G6:~/CC$ lex cal_084.l
suhami@LAPTOP-OTTF73G6:~/CC$ gcc y.tab.c lex.yy.c -o calssd -lfl
suhami@LAPTOP-OTTF73G6:~/CC$ ./calssd
Desk Calculator: Enter expressions (Ctrl+C to exit)
5+3
Result = 8
5+0/6
Result = 5
20/4
Result = 5
210/10+10
Result = 31
1+3-5/7*9
Result = 4
1+-5
Syntax Error! Please re-enter.
1**9
Syntax Error! Please re-enter.
1*9
Result = 9
^C
```

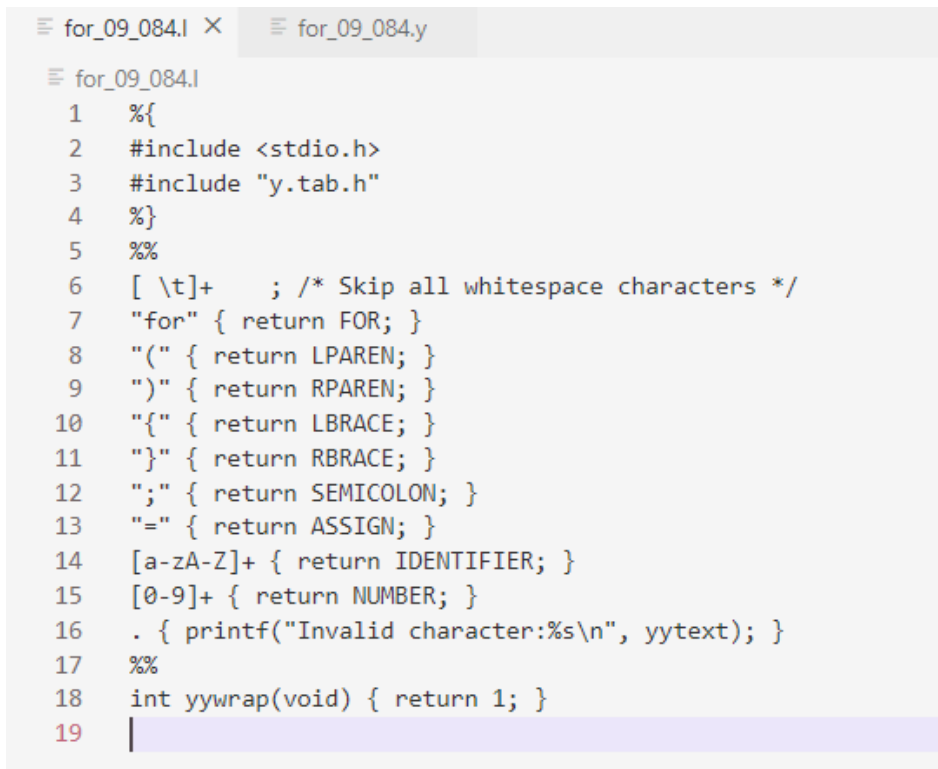
```
suhami@LAPTOP-OTTF73G6:~/CC$ yacc -d cal_084.y
suhami@LAPTOP-OTTF73G6:~/CC$ lex cal_084.l
suhami@LAPTOP-OTTF73G6:~/CC$ gcc y.tab.c lex.yy.c -o calssd -lfl
suhami@LAPTOP-OTTF73G6:~/CC$ ./calssd
Desk Calculator: Enter expressions (Ctrl+C to exit)
5+3
Result = 8
5+0/6
Result = 5
20/4
Result = 5
210/10+10
Result = 31
1+3-5/7*9
Result = 4
1+-5
Syntax Error! Please re-enter.
1**9
Syntax Error! Please re-enter.
1*9
Result = 9
^C
```

PRACTICAL 9

AIM - Parser for "FOR" loop statements.

file.l

```
%{
#include <stdio.h>
#include "y.tab.h"
%}
%%
[ \t]+ ; /* Skip all whitespace characters */
"for" { return FOR; }
"(" { return LPAREN; }
")" { return RPAREN; }
"{" { return LBRACE; }
"}" { return RBRACE; }
";" { return SEMICOLON; }
"=" { return ASSIGN; }
[a-zA-Z]+ { return IDENTIFIER; }
[0-9]+ { return NUMBER; }
. { printf("Invalid character:%s\n", yytext); }
%%
int yywrap(void) { return 1; }
```

A screenshot of a code editor with two tabs at the top: 'for_09_084.l' (active) and 'for_09_084.y'. The editor displays the content of 'for_09_084.l', which is the same as the code block above. The code is numbered from 1 to 19 on the left margin. The cursor is positioned at the end of line 19, which is a blank line. The editor has a light gray background and a dark border.

file.y

```
%{
#include <stdio.h>
#include <stdlib.h>
void yyerror(char *s);
```

```

int yylex(void);
%}
%token FOR LPAREN RPAREN LBRACE RBRACE SEMICOLON ASSIGN IDENTIFIER
NUMBER
%%
program:
for_loop
;
for_loop:
FOR LPAREN initialization SEMICOLON condition SEMICOLON update RPAREN LBRACE body
RBRACE
{
printf("Valid FOR loop structure\n");
}
;
initialization:
IDENTIFIER ASSIGN NUMBER
;
condition:
IDENTIFIER
;
update:
IDENTIFIER
;
body:
/* Empty for simplicity*/
;
%%
void yyerror(char *s) { fprintf(stderr, "Error: %s\n", s); }
int main(void) {
    yyparse();
    return 0;
}

```

```

≡ for_09_084.l  ≡ for_09_084.y X
≡ for_09_084.y
1  %{
2  #include <stdio.h>
3  #include <stdlib.h>
4  void yyerror(char *s);
5  int yylex(void);
6  %}
7  %token FOR LPAREN RPAREN LBRACE RBRACE SEMICOLON ASSIGN IDENTIFIER NUMBER
8  %%
9  program:
10 for_loop
11 ;
12 for_loop:
13 FOR LPAREN initialization SEMICOLON condition SEMICOLON update RPAREN LBRACE body RBRACE
14 {
15 printf("Valid FOR loop structure\n");
16 }
17 ;
18 initialization:
19 IDENTIFIER ASSIGN NUMBER
20 ;
21 condition:
22 IDENTIFIER
23 ;
24 update:
25 IDENTIFIER
26 ;
27 body:
28 /* Empty for simplicity*/
29 ;
30 %%
31 void yyerror(char *s) { fprintf(stderr, "Error: %s\n", s); }
32 int main(void) {
33     yyparse();
34     return 0;
35 }

```

```

suhami@LAPTOP-OTTF73G6:~/CC$ lex for_09_084.l
suhami@LAPTOP-OTTF73G6:~/CC$ gcc y.tab.c lex.yy.c -o for_parser -lfl
suhami@LAPTOP-OTTF73G6:~/CC$ ./for_parser
for(i=10; i; i) { }
Valid FOR loop structure

```

```

for(i=; i; i;) { }
Error: syntax error

```

```

suhami@LAPTOP-OTTF73G6:~/CC$ yacc -d for_09_084.y
suhami@LAPTOP-OTTF73G6:~/CC$ lex for_09_084.l
suhami@LAPTOP-OTTF73G6:~/CC$ gcc y.tab.c lex.yy.c -o for_parser -lfl
suhami@LAPTOP-OTTF73G6:~/CC$ ./for_parser
for(i=10; i; i) { }
Valid FOR loop structure

for(i=; i; i;) { }
Error: syntax error
suhami@LAPTOP-OTTF73G6:~/CC$ |

```

PRACTICAL 10

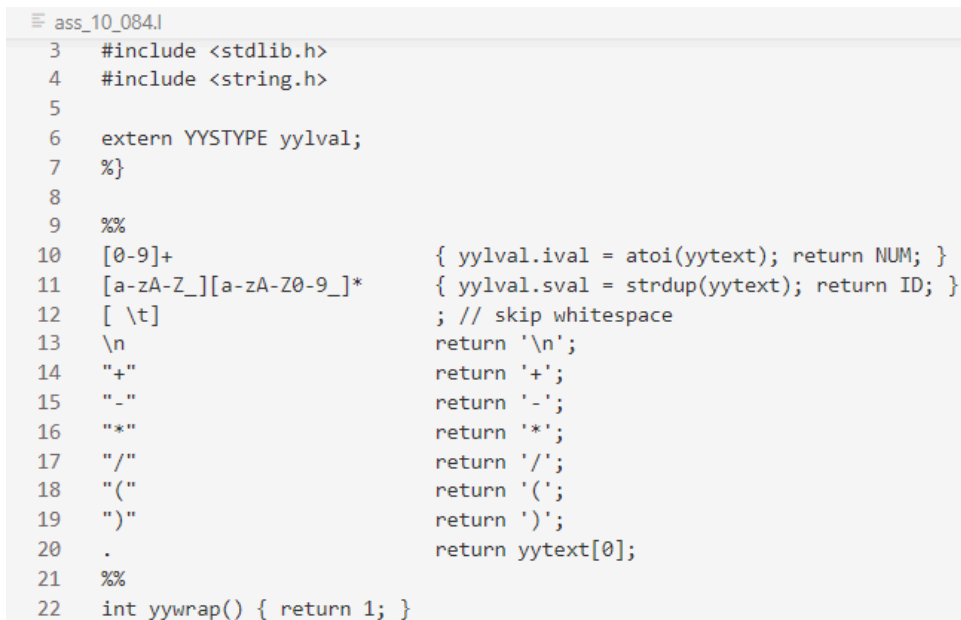
AIM - Intermediate code (IC) generator for arithmetic expression.

file.l

```
%{
#include "y.tab.h"
#include <stdlib.h>
#include <string.h>

extern YYSTYPE yylval;
%}

%%
[0-9]+      { yylval.ival = atoi(yytext); return NUM; }
[a-zA-Z_][a-zA-Z0-9_]* { yylval.sval = strdup(yytext); return ID; }
[ \t]       ; // skip whitespace
\n          return '\n';
"+"         return '+';
"_"         return '-';
"*"         return '*';
"/"         return '/';
"("         return '(';
")"         return ')';
.           return yytext[0];
%%
int yywrap() { return 1; }
```



```
3  #include <stdlib.h>
4  #include <string.h>
5
6  extern YYSTYPE yylval;
7  %{
8
9  %%
10 [0-9]+      { yylval.ival = atoi(yytext); return NUM; }
11 [a-zA-Z_][a-zA-Z0-9_]* { yylval.sval = strdup(yytext); return ID; }
12 [ \t]       ; // skip whitespace
13 \n          return '\n';
14 "+"         return '+';
15 "_"         return '-';
16 "*"         return '*';
17 "/"         return '/';
18 "("         return '(';
19 ")"         return ')';
20 .           return yytext[0];
21 %%
22 int yywrap() { return 1; }
```

file.y

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

```

int yylex(void); // Declare yylex to avoid implicit declaration warning
int tempCount = 1;
char* createTemp() {
    char* temp = (char*) malloc(10);
    sprintf(temp, "t%d", tempCount++);
    return temp;
}
void yyerror(const char* s) {
    fprintf(stderr, "Error: %s\n", s);
}
%}

%union {
    int ival;
    char* sval;
}
%token <ival> NUM
%token <sval> ID
%type <sval> expr
%left '+' '-'
%left '*' '/'

%%

stmt: expr '\n' { printf("\n"); };
expr: expr '+' expr {
    char* temp = createTemp();
    printf("%s = %s + %s\n", temp, $1, $3);
    $$ = temp;
}
| expr '-' expr {
    char* temp = createTemp();
    printf("%s = %s - %s\n", temp, $1, $3);
    $$ = temp;
}
| expr '*' expr {
    char* temp = createTemp();
    printf("%s = %s * %s\n", temp, $1, $3);
    $$ = temp;
}
| expr '/' expr {
    char* temp = createTemp();
    printf("%s = %s / %s\n", temp, $1, $3);
    $$ = temp;
}
| '(' expr ')' {
    $$ = $2;
}
| ID {
    $$ = $1;
}
| NUM {
    char* temp = (char*) malloc(10);
    sprintf(temp, "%d", $1);

```



```

    $$ = temp;
}
;

%%

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

```

```

≡ ass_10_084.y
1  %{
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5  int yylex(void); // Declare yylex to avoid implicit declaration warning
6  int tempCount = 1;
7  char* createTemp() {
8      char* temp = (char*) malloc(10);
9      sprintf(temp, "t%d", tempCount++);
10     return temp;
11 }
12 void yyerror(const char* s) {
13     fprintf(stderr, "Error: %s\n", s);
14 }
15 %}
16 %union {
17     int ival;
18     char* sval;
19 }
20 %token <ival> NUM
21 %token <sval> ID
22 %type <sval> expr
23 %left '+' '-'
24 %left '*' '/'
25 %%
26 stmt: expr '\n' { printf("\n"); };
27 expr: expr '+' expr {
28     char* temp = createTemp();
29     printf("%s = %s + %s\n", temp, $1, $3);
30     $$ = temp;
31 }
32 | expr '-' expr {
33     char* temp = createTemp();
34     printf("%s = %s - %s\n", temp, $1, $3);
35     $$ = temp;
36 }
37 | expr '*' expr {
38     char* temp = createTemp();
39     printf("%s = %s * %s\n", temp, $1, $3);
40     $$ = temp;
41 }

```

```

41     }
42     | expr '/' expr {
43         char* temp = createTemp();
44         printf("%s = %s / %s\n", temp, $1, $3);
45         $$ = temp;
46     }
47     | '(' expr ')' {
48         $$ = $2;
49     }
50     | ID {
51         $$ = $1;
52     }
53     | NUM {
54         char* temp = (char*) malloc(10);
55         sprintf(temp, "%d", $1);
56         $$ = temp;
57     }
58 ;
59 %%
60 int main() {
61     printf("Enter arithmetic expression:\n");
62     yyparse();
63     return 0;
64 }

```

```

suhami@LAPTOP-OTTF73G6:~/CC$ lex ass_10_084.l
suhami@LAPTOP-OTTF73G6:~/CC$ yacc -d ass_10_084.y
suhami@LAPTOP-OTTF73G6:~/CC$ gcc lex.yy.c y.tab.c -o ass
suhami@LAPTOP-OTTF73G6:~/CC$ ./ass

```

Enter arithmetic expression:

a+b*c

t1 = b * c

t2 = a + t1

ab+c*

Error: syntax error

suhami@LAPTOP-OTTF73G6:~/CC\$

```

suhami@LAPTOP-OTTF73G6:~/CC$ lex ass_10_084.l
suhami@LAPTOP-OTTF73G6:~/CC$ yacc -d ass_10_084.y
suhami@LAPTOP-OTTF73G6:~/CC$ gcc lex.yy.c y.tab.c -o ass
suhami@LAPTOP-OTTF73G6:~/CC$ ./ass
Enter arithmetic expression:
a+b*c
t1 = b * c
t2 = a + t1

ab+c*
Error: syntax error
suhami@LAPTOP-OTTF73G6:~/CC$ |

```

