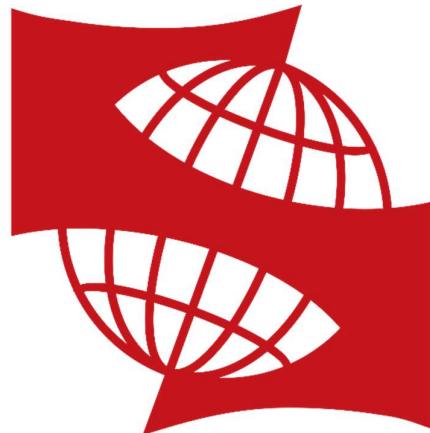


Symbiosis Institute of Technology, Nagpur



॥वसुधैव कुटुम्बकम्॥

**Computer Science and Engineering
Batch 2022-26**

**Course Name: Compiler Construction
Lab**

Name: Sujal Junghare

PRN: 22070521089

Division: C

Department: CSE

Semester-VII

Course Code: T7478

I. 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;
}
```

```
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ gedit number1089.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ lex number1089.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
ONE
1
zero
0
ZERO
0
□
```

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

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

int sc = 0, wc = 0, lc = 0, cc = 0;
int keyword_count = 0, identifier_count = 0, comment_count = 0, tab_count = 0;

char *keywords[] = {
"auto", "break", "case", "char", "const", "continue",
"default", "do", "double", "else", "enum", "extern",
"float", "for", "goto", "if", "inline", "int", "long", "register",
"restrict", "return", "short", "signed", "sizeof", "static", "struct",
"switch", "typedef", "union", "unsigned", "void", "volatile", "while"
};

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

%%

/*([^\*]|*+[^\*/])*/*+/* { comment_count++; cc += yyleng; }
//.* { comment_count++; cc += yyleng; }
[\n] { lc++; cc += yyleng; }
" " { sc++; cc += yyleng; }
\t { tab_count++; cc += yyleng; }
[^t\n ]+ {
wc++;
cc += yyleng;
if (is_keyword(yytext)) {
keyword_count++;
} else if (isalpha(yytext[0]) || yytext[0] == '_') {
```

```
    identifier_count++;
}
}
```

```
%%
```

```
int main(int argc, char* argv[]) {
printf("Enter the input:\n");
yylex();
printf("The number of lines = %d\n", lc);
printf("The number of spaces = %d\n", sc);
printf("The number of tabs = %d\n", tab_count);
printf("The number of words = %d\n", wc);
printf("The number of characters = %d\n", cc);
printf("The number of keywords = %d\n", keyword_count);
printf("The number of identifiers = %d\n", identifier_count);
printf("The number of comments = %d\n", comment_count);
return 0;
}
```

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

```
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ gedit number21089.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ lex number21089.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 sentence : I am Blaze and I am Studying.
Symbiosis Institute of Technology, Nagpur.
Number of lines : 2
Number of spaces : 10
Number of tabs, words, charc : 2 , 10 , 73
```

III. Count number of words starting with 'A'.

```
%{
#include <stdio.h>
int cap_a_count = 0;
int small_a_count = 0;
%}

%%%  
A[a-zA-Z0-9_]* { cap_a_count++; }
a[a-zA-Z0-9_]* { small_a_count++; }
[^ \t\n]+ /* Ignore other words */
[ \t\n]+ /* Skip spaces */

%%%  
  
int main()
{
    printf("Enter the input text (Ctrl+D to end):\n");
    yylex();
    printf("\nNumber of words starting with 'A' = %d\n", cap_a_count);
    printf("Number of words starting with 'a' = %d\n", small_a_count);
    printf("Total words starting with 'A' or 'a' = %d\n", cap_a_count +
small_a_count);
    return 0;
}  
  
int yywrap()
{
return 1;
}  
  
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ gedit count21089.l  
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ lex count21089.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 text : I am Sujal Junghare.  
  
Number of words starting with 'A' or 'a': 3
```

IV. Conversion of lowercase to uppercase and vice versa.

```
%{  
#include <stdio.h>  
#include <ctype.h>  
%}  
  
%%  
  
[a-z] { printf("%c", toupper(yytext[0])); }  
[A-Z] { printf("%c", tolower(yytext[0])); }  
. { printf("%s", yytext); }  
  
%%
```

```
int main() {  
printf("Enter the input text :\n");  
yylex();  
return 0;  
}
```

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

```
blaze@BLAZE:~$ lex practical4_1089.l  
blaze@BLAZE:~$ cc lex.yy.c -ll  
blaze@BLAZE:~$ ./a.out  
Sujal Junghare  
sUJAL jUNGHARE
```

V. 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 = num / 16;
    }

    printf("0x");
    int j;
    for(j = i - 1; j >= 0; j--)
        printf("%c", hex[j]);
    printf("\n");
}

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]+ {
    int num = string_to_int(yytext);
```

```
        decimal_to_hex(num);  
    }  
  
.\\n { ECHO; }  
%%
```

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

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

```
blaze@BLAZE:~$ lex practical5_1089.l  
blaze@BLAZE:~$ cc lex.yy.c -ll  
blaze@BLAZE:~$ ./a.out  
Enter decimal number (ctrl+d to end):  
935  
0x3A7  
  
093  
0x5D  
  
875842  
0xD5D42  
  
11111  
0x2B67
```

VI. Test lines ending with "com".

```
%{
#include <stdio.h>
int line_number = 1;
%}

%%*
.com\n {
    printf("Line %d ends with 'com': %s", line_number, yytext);
    line_number++;
}
\n {
    line_number++;
}
.\t /* Consume other characters */
%%

int main() {
    yylex();
    return 0;
}

int yywrap() {
    return 1;
}

blaze@BLAZE:~$ lex practical6_1089.l
blaze@BLAZE:~$ cc lex.yy.c -ll
blaze@BLAZE:~$ ./a.out
Enter text (Ctrl+D to end input):
sujal.r.junghare@gmail.com
Line ends with 'com': sujal.r.junghare@gmail.com
blaze@BLAZE:~$
```

VII. Postfix Expression Evaluation.

Lex File:

```
%{  
#include <stdio.h>  
#include <stdlib.h>  
#include "y.tab.h"  
%}  
  
%%%  
  
[0-9] { yyval = yytext[0] - '0'; return NUMBER; }  
[ \t] ; /* ignore spaces/tabs */  
\n { return '\n'; } /* pass newline to parser */  
. { return yytext[0]; } /* operators like + - * / */  
  
%%%
```

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

Yacc File:

```
%{  
#include <stdio.h>  
#include <stdlib.h>  
int yylex(void);  
void yyerror(const char *s);  
%}  
  
%token NUMBER  
  
%%%  
  
input  
: /* empty */  
| input line  
;  
  
line  
: expr '\n' { printf("Result = %d\n", $1); }  
| '\n' /* empty line */  
;  
  
expr  
: NUMBER  
| expr expr '+' { $$ = $1 + $2; }
```

```

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

%%%

```

```

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

```

```

int main(void) {
    printf("Enter postfix expressions:\n");
    return yyparse();
}

```

```

sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ gedit postfix_1089.y
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ gedit postfix_1089.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ yacc -d postfix_1089.y
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ lex postfix_1089.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ gcc lex.yy.c y.tab.c -o postfix -ll
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ ./postfix
Enter postfix expressions:
23*54*+9-
Result = 17

```

VIII. Desk calculator with error recovery.

Lex File:

```
%{  
#include "y.tab.h"  
%}  
  
digit [0-9]  
  
%%%  
  
[ \t]+ ; // ignore whitespace  
{digit}+ { yyval = atoi(yytext); return NUMBER; }  
[\n] { return '\n'; }  
 "+" { return '+'; }  
 "-" { return '-'; }  
 "*" { return '*' };  
 "/" { return '/'; }  
 "(" { return '('; }  
 ")" { return ')'; }  
. { printf("Unknown character: %s\n", yytext); }
```

```
%%%
```

Yacc File:

```
%{  
#include <stdio.h>  
#include <stdlib.h>  
  
void yyerror(const char *s);  
int yylex(void);  
  
%}  
  
%token NUMBER  
  
%left '+' '-'  
%left '*' '/'  
%right UMINUS  
  
%%%  
input:  
/* empty */  
| input line  
;
```

```

line:
'\n'
| expr '\n' { printf("Result = %d\n", $1); }
| error '\n' { yyerror("Syntax error recovered."); yyclearin; }
;

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

%%

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

int main(void) {
    printf("Enter expressions:\n");
    return yyparse();
}

blaze@BLAZE:~$ yacc -d practical8_1089.y
blaze@BLAZE:~$ lex practical8_1089.l
blaze@BLAZE:~$ gcc -o calc lex.yy.c y.tab.c -ll
blaze@BLAZE:~$ ./calc
Simple Desk Calculator (type expressions or Ctrl+D to exit)
9+3
=12
100/5
=20

```

IX. Parser for "FOR" loop statements.4

Lex file:

```
%{  
#include "y.tab.h"  
%}  
  
%%  
  
"for"           return FOR;  
"int"           return INT;  
"float"         return FLOAT;  
"char"          return CHAR;  
"double"        return DOUBLE;  
  
 "("            return LPAREN;  
 ")"            return RPAREN;  
 ";"             return SEMICOLON;  
 "{"            return LBRACE;  
 "}"            return RBRACE;  
  
 "++"           return INC;  
 "--"           return DEC;  
 "<="           return LE;  
 ">="           return GE;  
 "=="           return EQ;  
 "!="           return NE;  
 "<"            return LT;  
 ">"            return GT;  
 "="             return ASSIGN;  
 "+"             return PLUS;  
 "-"             return MINUS;  
 "*"             return MULT;  
 "/"             return DIV;  
  
[a-zA-Z_][a-zA-Z0-9_]* return ID;  
[0-9]+          { yyval.num = atoi(yytext); return NUMBER; }  
  
[ \t\r\n]+        ; /* ignore whitespace */  
.              return yytext[0];  
  
%%  
  
int yywrap() { return 1; }
```

Yacc File:

```
%{

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

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

%union {
    int num;
    char *str;
}

%token <num> NUMBER
%token ID
%token FOR LPAREN RPAREN SEMICOLON ASSIGN
%token LT GT LE GE EQ NE
%token PLUS MINUS MULT DIV
%token INC DEC
%token LBRACE RBRACE
%token INT FLOAT CHAR DOUBLE

%left PLUS MINUS
%left MULT DIV

%%

program:
    for_stmt
;

for_stmt:
    FOR LPAREN init SEMICOLON condition SEMICOLON increment
    RPAREN body
    { printf("Valid FOR loop statement parsed.\n"); }
;

init:
    datatype ID ASSIGN expr
    | ID ASSIGN expr
;
```

datatype:

```
    INT  
    | FLOAT  
    | CHAR  
    | DOUBLE  
    ;
```

condition:

```
    expr LT expr  
    | expr GT expr  
    | expr LE expr  
    | expr GE expr  
    | expr EQ expr  
    | expr NE expr  
    ;
```

increment:

```
    ID ASSIGN expr  
    | ID INC  
    | ID DEC  
    | INC ID  
    | DEC ID  
    ;
```

body:

```
    statement  
    | LBRACE statements RBRACE  
    ;
```

statements:

```
    /* empty */  
    | statements statement  
    ;
```

statement:

```
    expr SEMICOLON  
    | for_stmt  
    ;
```

expr:

```
    NUMBER  
    | ID
```

```

| expr PLUS expr
| expr MINUS expr
| expr MULT expr
| expr DIV expr
| ID INC
| ID DEC
| INC ID
| DEC ID
;

%%

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

int main() {
    printf("Enter a FOR loop statement:\n");
    yyparse();
    return 0;
}

```

```

blaze@BLAZE:~$ lex practical9_1089.l
blaze@BLAZE:~$ yacc -d practical9_1089.y
blaze@BLAZE:~$ gcc lex.yy.c y.tab.c -o prac9
blaze@BLAZE:~$ ./prac9
for(i=0; i; i) {}
Valid FOR loop structure

for(i=0; i<5; i++) {}
Error: syntax error

```

X. Intermediate code (IC) generator for arithmetic expression.

Lex File:

```
%{  
#include "y.tab.h"  
#include <stdlib.h>  
#include <string.h>  
%}  
  
%%  
  
[0-9]+ { yyval.str = strdup(yytext); return NUMBER; }  
[a-zA-Z_][a-zA-Z0-9_]* { yyval.str = strdup(yytext); return ID; }  
  
[\t ]+ ; /* skip whitespace */  
\n { return '\n'; }  
  
"+" { return '+'; }  
"-" { return '-'; }  
"*" { return '*'; }  
"/" { return '/'; }  
  
. { return yytext[0]; }  
  
%%
```

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

Yacc File:

```
%{  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
  
int tempCount = 0;  
int yylex(void);  
void yyerror(const char *s);  
%}  
  
%union {  
    char *str;  
}  
  
%token <str> ID NUMBER
```

```

%left '+' '-'
%left '*' '/'
%type <str> expr

%%

input:
/* empty */
| input expr '\n' { printf("\n"); free($2); }
;

expr:
expr '+' expr {
    $$ = (char *)malloc(20);
    sprintf($$, "t%od", ++tempCount);
    printf("%s = %s + %s\n", $$, $1, $3);
    free($1); free($3);
}
| expr '-' expr {
    $$ = (char *)malloc(20);
    sprintf($$, "t%od", ++tempCount);
    printf("%s = %s - %s\n", $$, $1, $3);
    free($1); free($3);
}
| expr '*' expr {
    $$ = (char *)malloc(20);
    sprintf($$, "t%od", ++tempCount);
    printf("%s = %s * %s\n", $$, $1, $3);
    free($1); free($3);
}
| expr '/' expr {
    $$ = (char *)malloc(20);
    sprintf($$, "t%od", ++tempCount);
    printf("%s = %s / %s\n", $$, $1, $3);
    free($1); free($3);
}
| ID   { $$ = strdup($1); free($1); }
| NUMBER { $$ = strdup($1); free($1); }
;
%%
```

```

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

```
}
```

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

```
blaze@BLAZE:~$ lex practical10_1089.l
blaze@BLAZE:~$ yacc -d practical10_1089.y
blaze@BLAZE:~$ gcc lex.yy.c y.tab.c -o prac10
blaze@BLAZE:~$ ./prac10
Enter arithmetic expression:
a + b / c * d
t1 = b / c
t2 = t1 * d
t3 = a + t2
```