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)

॥वसुधैव कुटुम्बकम्॥ 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
~/

```
%{
#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;
}
```

```
^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;
}
```

File: /home/sit/file_084.l Page 1 of 1

```
%{
#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;
}
```

```
%{
#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;
}
```

```
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$
```

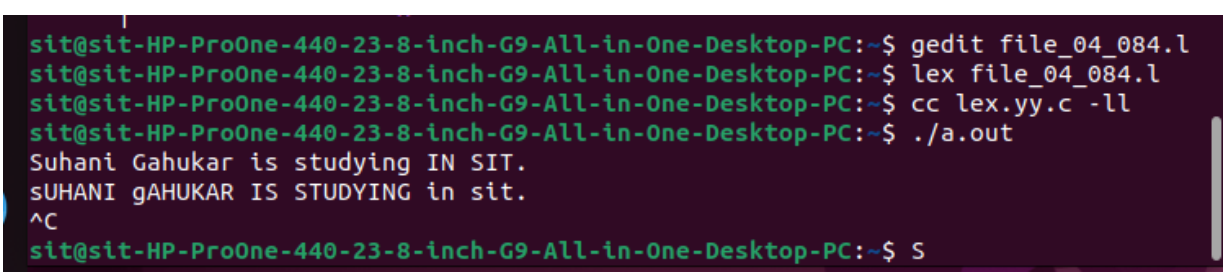
PRACTICAL 4

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;
}
```



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

prm5_084.l
~/

```
%{
#include <stdio.h>
%}
%%
[0-9]+  {
            int decimal_value = 0;
            char hex_string[100]; // Buffer to store the hexadecimal digits
            int i = 0;

            // Manual conversion from string to integer
            for (int k = 0; yytext[k] != '\0'; k++) {
                decimal_value = decimal_value * 10 + (yytext[k] - '0');
            }

            // Convert and print hexadecimal
            if (decimal_value == 0) {
                printf("0x0\n");
            } else {
                while (decimal_value > 0) {
                    int remainder = decimal_value % 16;
                    if (remainder < 10) {
                        hex_string[i] = remainder + '0';
                    } else {
                        hex_string[i] = remainder - 10 + 'A';
                    }
                    decimal_value = decimal_value / 16;
                    i++;
                }

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

.|\n    {
            putchar(yytext[0]);
        }

%%

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

```
prm5_084.l:3: unrecognized rule
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ gedit prm5_084.l
sit@sit-HP-ProOne-440-23-8-inch-G9-All-in-One-Desktop-PC:~$ lex prm5_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

```
file_6_084.l
%{
#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;
}
```

```
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

```
%{
#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); }
```

postfix_084.l postfix_084.y

postfix_084.y

```
element:
    | '+'                    {
                              }
    | '-'                    {
                              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;
}
```

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 × postfix_084.y

postfix_084.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); }
%%
```

```
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;
}
```

Welcome | cal_084.l | cal_084.y

cal_084.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"); yyerrok; }
      ;
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

cal_084.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"); yyerrok; }
      ;

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
}
```

```
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; }
```

for_09_084.l × for_09_084.y

for_09_084.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; }
```

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

for_09_084.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;
}
```

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; }
```

ass_10_084.l

```
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

```
%{
#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;
}
```

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$