**Practical – 9**

**Aim:**

**(a) Write a YACC program to recognize a valid arithmetic expression for operators (+,-,\*, /).**

**(b) Write a YACC program to check syntax of for loop.**

**Code (a):**

**practical9a.l**

%{

#include "y.tab.h"

extern int yylval;

%}

%%

[0-9]+ {

/\* convert yytext to an integer \*/

yylval = atoi(yytext);

return NUM;

}

[ \t] /\* Ignore whitespaces \*/;

\n return 0;

. return yytext[0];

%%

**practical9a.y**

%{

#include <stdio.h>

void yyerror(char \*s);

int yylex(void);

%}

%token NUM

%left '+' '-' '\*' '/'

%%

S: EXPR { printf("Valid arithmetic expression\n"); }

EXPR: EXPR '+' EXPR

| EXPR '-' EXPR

| EXPR '\*' EXPR

| EXPR '/' EXPR

| NUM

;

%%

void yyerror(char \*s) {

fprintf(stderr, "Invalid expression (%s: detected)\n", s);

}

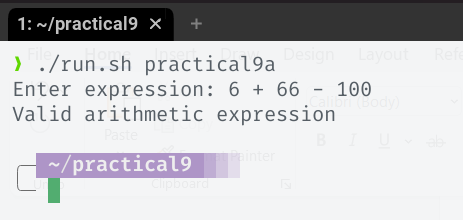
int main() {

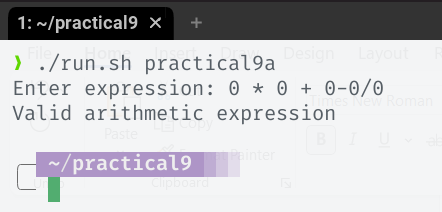
printf("Enter expression: ");

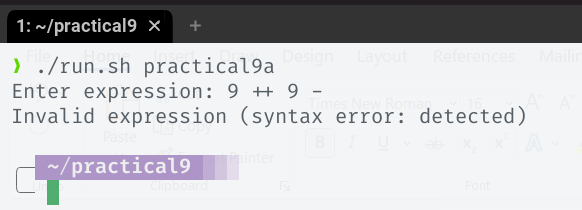
return yyparse();

}

**Output (a):**

****

****

****

**Code (b):**

**practical9b.l**

%{

#include "y.tab.h"

%}

NUMBER [0-9]

LETTER [a-zA-Z\_]

%%

"for" return FOR;

"(" return LPAREN;

")" return RPAREN;

"=" return ASSIGN;

"=="|"<="|">="|"<"|">" return RELOP;

"int "|"float "|"char " return TYPE;

{LETTER}({LETTER}|{NUMBER})\* return ID;

";" return SEMI;

{NUMBER}+ return NUMBER;

"++"|"--"|"+="|"-=" return INCDEC;

[ \t] /\* ignore \*/;

"\n" return NEWLINE;

. return yytext[0];

%%

**practical9b.y**

%{

#include <stdio.h>

void yyerror(char \*s);

int yylex(void);

%}

%token FOR LPAREN RPAREN ID TYPE NUMBER RELOP INCDEC ASSIGN SEMI NEWLINE

%%

SD: S NEWLINE { printf("for loop prototype is correct\n"); return 0; }

S: FOR LPAREN INNER RPAREN;

INNER: T ID ASSIGN NUMBER SEMI ID RELOP NUMBER SEMI ID OPR

|

SEMI SEMI

;

T: TYPE

|

;

OPR: INCDEC

|

INCDEC NUMBER

;

%%

void yyerror(char \*s)

{

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

}

int main(void)

{

yyparse();

return 0;

}

**Output (b):**

