Symbiosis Institute of Technology, Pune

Faculty of Engineering

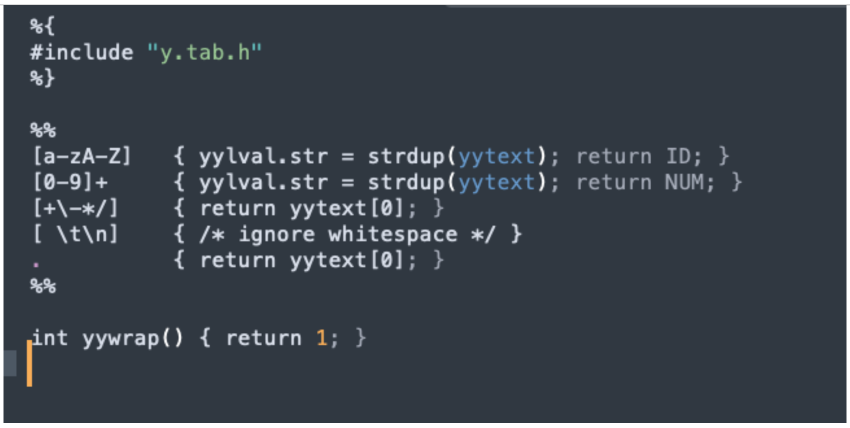
CSE- Academic Year 2025-26

Compiler Construction Lab   Batch 2022-26

Lab Assignment No: - 10	
Name of Student	Soham Phadke
PRN No.	22070122214
Batch	2022-2026
Class	C2
Academic Year & Semester	2025, 7th sem
Title of Assignment	Parser for Intermediate code (IC) generator for arithmetic expression.
Practice Questions	1. YACC program for Intermediate code (IC) generator for arithmetic expression. 2. YACC program for IC generation for the expression involving parenthesis.
Source Code	1.

Icgen.l

```
%{
#include "y.tab.h"
%}

%%
[a-zA-Z]   { yylval.str = strdup(yytext); return ID; }
[0-9]+     { yylval.str = strdup(yytext); return NUM; }
[+\-*/]    { return yytext[0]; }
[ \t\n]    { /* ignore whitespace */ }
.          { return yytext[0]; }
%%

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

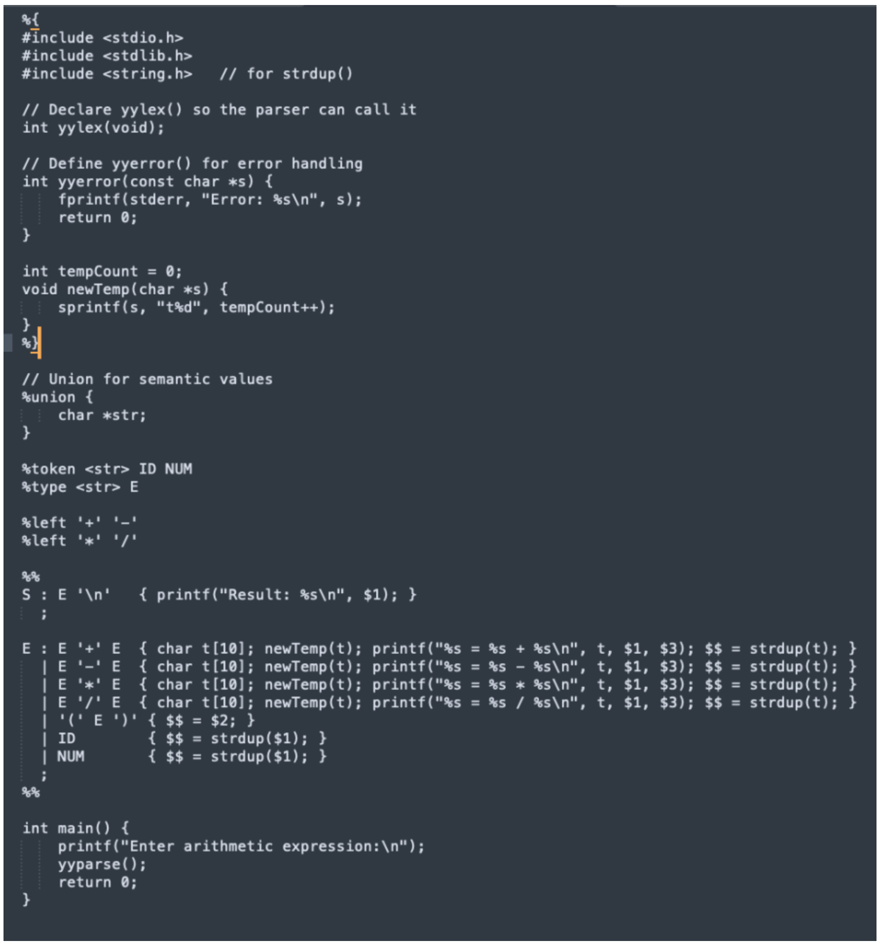


```
%{
#include "y.tab.h"
%}

%%
[a-zA-Z]   { yylval.str = strdup(yytext); return ID; }
[0-9]+     { yylval.str = strdup(yytext); return NUM; }
[+\-*/]    { return yytext[0]; }
[ \t\n]    { /* ignore whitespace */ }
.          { return yytext[0]; }
%%

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

icgen.y

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>   // for strdup()


int yylex(void);

int yyerror(const char *s) {
    fprintf(stderr, "Error: %s\n", s);
    return 0;
}

int tempCount = 0;
void newTemp(char *s) {
    sprintf(s, "t%d", tempCount++);
}
%}

%union {
    char *str;
}

%token <str> ID NUM
%type <str> E

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

%%
S : E '\n'   { printf("Result: %s\n", $1); }
  ;
```

```
E : E '+' E  { char t[10]; newTemp(t); printf("%s = %s + %s\n", t, $1, $3); $$ = strdup(t); }
  | E '-' E  { char t[10]; newTemp(t); printf("%s = %s - %s\n", t, $1, $3); $$ = strdup(t); }
  | E '*' E  { char t[10]; newTemp(t); printf("%s = %s * %s\n", t, $1, $3); $$ = strdup(t); }
  | E '/' E  { char t[10]; newTemp(t); printf("%s = %s / %s\n", t, $1, $3); $$ = strdup(t); }
  | '(' E ')' { $$ = $2; }
  | ID        { $$ = strdup($1); }
  | NUM       { $$ = strdup($1); }
  ;
%%

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

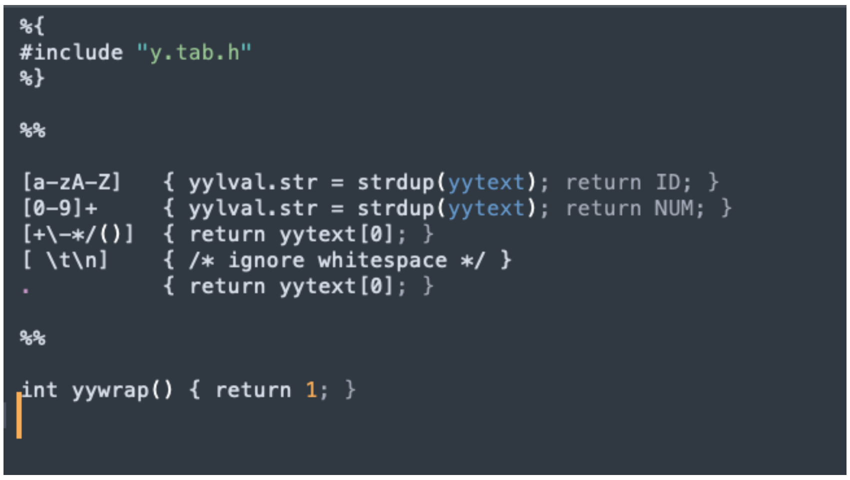


```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>   // for strdup()

// Declare yylex() so the parser can call it
int yylex(void);

// Define yyerror() for error handling
int yyerror(const char *s) {
    fprintf(stderr, "Error: %s\n", s);
    return 0;
}

int tempCount = 0;
void newTemp(char *s) {
    sprintf(s, "t%d", tempCount++);
}
%}

// Union for semantic values
%union {
    char *str;
}

%token <str> ID NUM
%type <str> E

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

%%
S : E '\n'   { printf("Result: %s\n", $1); }
  ;

E : E '+' E  { char t[10]; newTemp(t); printf("%s = %s + %s\n", t, $1, $3); $$ = strdup(t); }
  | E '-' E  { char t[10]; newTemp(t); printf("%s = %s - %s\n", t, $1, $3); $$ = strdup(t); }
  | E '*' E  { char t[10]; newTemp(t); printf("%s = %s * %s\n", t, $1, $3); $$ = strdup(t); }
  | E '/' E  { char t[10]; newTemp(t); printf("%s = %s / %s\n", t, $1, $3); $$ = strdup(t); }
  | '(' E ')' { $$ = $2; }
  | ID        { $$ = strdup($1); }
  | NUM       { $$ = strdup($1); }
  ;
%%

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

2.

ic_paren.l

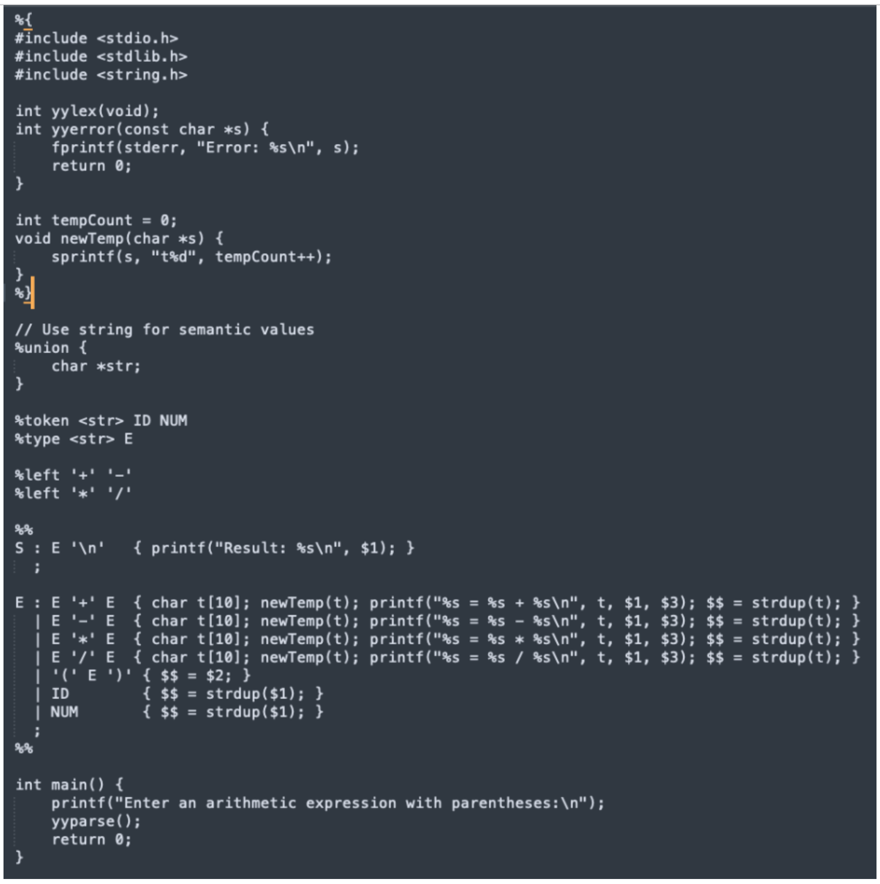
```
%{
#include "y.tab.h"
%}

%%

[a-zA-Z]   { yylval.str = strdup(yytext); return ID; }
[0-9]+     { yylval.str = strdup(yytext); return NUM; }
[+\-*/()]  { return yytext[0]; }
[ \t\n]    { /* ignore whitespace */ }
.          { return yytext[0]; }

%%

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



```
%{
#include "y.tab.h"
%}

%%

[a-zA-Z]   { yylval.str = strdup(yytext); return ID; }
[0-9]+     { yylval.str = strdup(yytext); return NUM; }
[+\-*/()]  { return yytext[0]; }
[ \t\n]    { /* ignore whitespace */ }
.          { return yytext[0]; }

%%

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

ic_paren.y

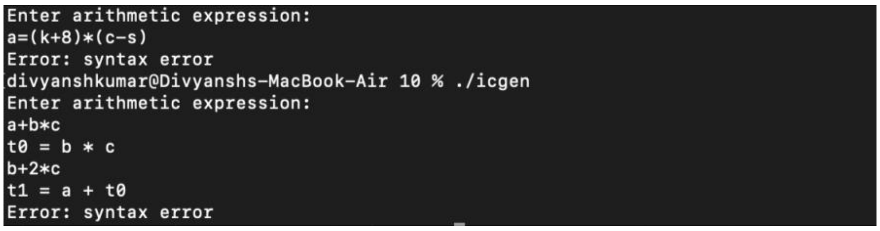
```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int yylex(void);
int yyerror(const char *s) {
    fprintf(stderr, "Error: %s\n", s);
    return 0;
}

int tempCount = 0;
void newTemp(char *s) {
    sprintf(s, "t%d", tempCount++);
}
%}

// Use string for semantic values
%union {
    char *str;
}

%token <str> ID NUM
%type <str> E

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

%%
S : E '\n'   { printf("Result: %s\n", $1); }
  ;

E : E '+' E  { char t[10]; newTemp(t); printf("%s = %s + %s\n", t, $1, $3); $$ = strdup(t); }
  | E '-' E  { char t[10]; newTemp(t); printf("%s = %s - %s\n", t, $1, $3); $$ = strdup(t); }
  | E '*' E  { char t[10]; newTemp(t); printf("%s = %s * %s\n", t, $1, $3); $$ = strdup(t); }
  | E '/' E  { char t[10]; newTemp(t); printf("%s = %s / %s\n", t, $1, $3); $$ = strdup(t); }
  | '(' E ')' { $$ = $2; }
  | ID       { $$ = strdup($1); }
  | NUM      { $$ = strdup($1); }
  ;
%%

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



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

int yylex(void);
int yyerror(const char *s) {
    fprintf(stderr, "Error: %s\n", s);
    return 0;
}

int tempCount = 0;
void newTemp(char *s) {
    sprintf(s, "t%d", tempCount++);
}
%}

// Use string for semantic values
%union {
    char *str;
}

%token <str> ID NUM
%type <str> E

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

%%
S : E '\n'   { printf("Result: %s\n", $1); }
  ;

E : E '+' E  { char t[10]; newTemp(t); printf("%s = %s + %s\n", t, $1, $3); $$ = strdup(t); }
  | E '-' E  { char t[10]; newTemp(t); printf("%s = %s - %s\n", t, $1, $3); $$ = strdup(t); }
  | E '*' E  { char t[10]; newTemp(t); printf("%s = %s * %s\n", t, $1, $3); $$ = strdup(t); }
  | E '/' E  { char t[10]; newTemp(t); printf("%s = %s / %s\n", t, $1, $3); $$ = strdup(t); }
  | '(' E ')' { $$ = $2; }
  | ID        { $$ = strdup($1); }
  | NUM       { $$ = strdup($1); }
  ;
%%

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

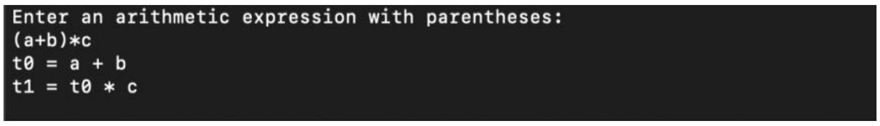
Output Screenshot

1.



```
Enter arithmetic expression:
a=(k+8)*(c-s)
Error: syntax error
divyanshkumar@Divyanshs-MacBook-Air 10 % ./icgen
Enter arithmetic expression:
a+b*c
t0 = b * c
b+2*c
t1 = a + t0
Error: syntax error
```

2.



```
Enter an arithmetic expression with parentheses:
(a+b)*c
t0 = a + b
t1 = t0 * c
```

Post lab questions

1. YACC program for IC generation for the expression involving programming constructs.

Conclusion

Ic_prog.y

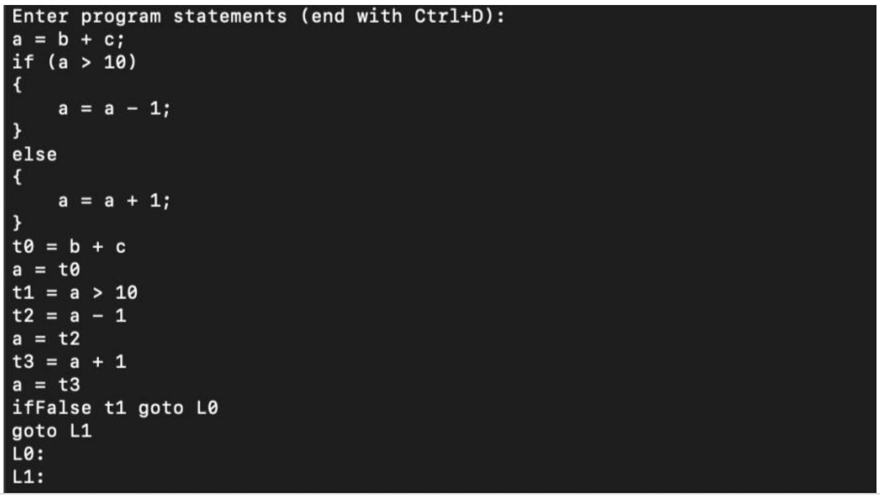
```
%{


#include <stdio.h>


#include <stdlib.h>


#include <string.h>


int yylex(void);
int yyerror(const char *s) {
fprintf(stderr, "Error: %s\n", s);
return 0;
}

int tempCount = 0;
int labelCount = 0;

void newTemp(char *s) {
sprintf(s, "t%d", tempCount++);
}

void newLabel(char *s) {
sprintf(s, "L%d", labelCount++);
}
%}

// Union for semantic values
%union {
char *str;
}

%token <str> ID NUM
%token IF ELSE WHILE
%token EQ NE LE GE

%type <str> E COND

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

%%
program: stmt_list
;
```

```
stmt_list: stmt_list stmt
| stmt
;

stmt: ID '=' E ';'   { printf("%s = %s\n", $1, $3); }
| IF '(' COND ')' stmt ELSE stmt
{
char L1[10], L2[10];
newLabel(L1); newLabel(L2);
printf("ifFalse %s goto %s\n", $3, L1);
/* stmt IC is printed inside */
printf("goto %s\n", L2);
printf("%s:\n", L1);
printf("%s:\n", L2);
}
| IF '(' COND ')' stmt
{
char L1[10];
newLabel(L1);
printf("ifFalse %s goto %s\n", $3, L1);
/* stmt IC is printed inside */
printf("%s:\n", L1);
}
| WHILE '(' COND ')' stmt
{
char L1[10], L2[10];
newLabel(L1); newLabel(L2);
printf("%s:\n", L1);
printf("ifFalse %s goto %s\n", $3, L2);
/* stmt IC printed inside */
printf("goto %s\n", L1);
printf("%s:\n", L2);
}
| '{' stmt_list '}'  { /* IC already printed */ }
;

E: E '+' E { char t[10]; newTemp(t); printf("%s = %s + %s\n", t, $1, $3); $$ = strdup(t); }
| E '-' E { char t[10]; newTemp(t); printf("%s = %s - %s\n", t, $1, $3); $$ = strdup(t); }
| E '*' E { char t[10]; newTemp(t); printf("%s = %s * %s\n", t, $1, $3); $$ = strdup(t); }
| E '/' E { char t[10]; newTemp(t); printf("%s = %s / %s\n", t, $1, $3); $$ = strdup(t); }
| '(' E ')' { $$ = $2; }
| ID { $$ = strdup($1); }
| NUM { $$ = strdup($1); }
;

COND: E EQ E { char t[10]; newTemp(t); printf("%s = %s == %s\n", t, $1, $3); $$ = strdup(t); }
```

```
| E NE E { char t[10]; newTemp(t); printf("%s = %s != %s\n", t, $1, $3); $$ = strdup(t); }
| E '<' E { char t[10]; newTemp(t); printf("%s = %s < %s\n", t, $1, $3); $$ = strdup(t); }
| E '>' E { char t[10]; newTemp(t); printf("%s = %s > %s\n", t, $1, $3); $$ = strdup(t); }
| E LE E { char t[10]; newTemp(t); printf("%s = %s <= %s\n", t, $1, $3); $$ = strdup(t); }
| E GE E { char t[10]; newTemp(t); printf("%s = %s >= %s\n", t, $1, $3); $$ = strdup(t); }
;
%%

int main() {
printf("Enter program statements (end with Ctrl+D):\n");
yyparse();
return 0;
}
```

ic_prog.l

```
%{
#include "y.tab.h"
%}

%%
"if"        { return IF; }
"else"      { return ELSE; }
"while"     { return WHILE; }
[a-zA-Z]    { yylval.str = strdup(yytext); return ID; }
[0-9]+      { yylval.str = strdup(yytext); return NUM; }
"="         { return '='; }
"=="        { return EQ; }
"!="        { return NE; }
"<="        { return LE; }
">="        { return GE; }
"<"         { return '<'; }
">"         { return '>'; }
[+\-*/()]   { return yytext[0]; }
[{};]       { return yytext[0]; }
[ \t\n]     { /* ignore whitespace */ }
.           { return yytext[0]; }
%%

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



```
Enter program statements (end with Ctrl+D):
a = b + c;
if (a > 10)
{
    a = a - 1;
}
else
{
    a = a + 1;
}
t0 = b + c
a = t0
t1 = a > 10
t2 = a - 1
a = t2
t3 = a + 1
a = t3
ifFalse t1 goto L0
goto L1
L0:
L1:
```