Symbiosis Institute of Technology, Pune

Faculty of Engineering

CSE- Academic Year 2025-26

Compiler Construction Lab Batch 2022-26

Lab Assignment No: - 9

Name: Soham Phadke

PRN: 22070122214

Batch: 2022-26

Class: CSE C2

Semester – 7th

Title of Assignment: Parser for "FOR" loop statements. Practice Questions

1. YACC program for parser for "FOR" loop statements.
2. YACC program for checking syntax for a While loop.
3. YACC program for checking syntax for a Switch case.

Q1.

Source Code

Lex file:

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

%%

"for"           { return FOR; }
"int"           { return INT; }
"("             { return LPAREN; }
")"             { return RPAREN; }
";"             { return SEMICOLON; }
"="             { return ASSIGN; }
"<="            { return LE; }
">="            { return GE; }
"=="            { return EQ; }
"!="            { return NE; }
"<"             { return LT; }
">"             { return GT; }
"++"            { return INC; }
"--"            { return DEC; }
[0-9]+          { yylval = atoi(yytext); return NUMBER; }
[a-zA-Z_][a-zA-Z0-9_]* { return ID; }
[ \t\n\r]+      ;
.               { return yytext[0]; }

%%

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

.Y file:

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

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

```
%token FOR INT LPAREN RPAREN SEMICOLON ASSIGN LT GT LE GE EQ NE INC DEC ID
NUMBER

%%
program:
    for_statement
  ;

for_statement:
    FOR LPAREN init SEMICOLON condition SEMICOLON increment RPAREN statement
     { printf("Valid FOR loop syntax\n"); }
  ;

init:
    INT ID ASSIGN NUMBER
  | ID ASSIGN NUMBER
  ;

condition:
    ID LT NUMBER
  | ID GT NUMBER
  | ID LE NUMBER
  | ID GE NUMBER
  | ID EQ NUMBER
  | ID NE NUMBER
  ;

increment:
    ID INC
  | ID DEC
  | ID ASSIGN ID
  | ID ASSIGN ID INC
  | ID ASSIGN ID DEC
  | ID ASSIGN ID '+' NUMBER
  | ID ASSIGN ID '-' NUMBER
  ;

statement:
    ID LPAREN ID RPAREN SEMICOLON
```

```
    | ID SEMICOLON
    ;
%%

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

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

Output Screenshot

```
PS C:\Users\Soham\Documents\SEM7\Sem7 codes> ./for_loop_parser
Enter a FOR loop statement:
for(int i=0; i<10; i++) printf(i);
Valid FOR loop syntax

for(i<10);
 Syntax error: syntax error
```

Q2.

Source Code

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

%%

"while"          { return WHILE; }
"int"            { return INT; }
"("              { return LPAREN; }
```

```
")"             { return RPAREN; }
"{"             { return LBRACE; }
"}"             { return RBRACE; }
";"             { return SEMICOLON; }
"="             { return ASSIGN; }
"<="            { return LE; }
">="            { return GE; }
"=="            { return EQ; }
"!="            { return NE; }
"<"             { return LT; }
">"             { return GT; }
"++"            { return INC; }
"--"            { return DEC; }
"+"             { return PLUS; }
"-"             { return MINUS; }
"*"             { return MULT; }
"/"             { return DIV; }
[0-9]+          { yylval = atoi(yytext); return NUMBER; }
[a-zA-Z_][a-zA-Z0-9_]* { return ID; }
[ \t\n\r]+      ;
.               { return yytext[0]; }

%%

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

.Y file:

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

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

%token WHILE INT LPAREN RPAREN LBRACE RBRACE SEMICOLON ASSIGN LT GT LE GE EQ NE INC DEC PLUS MINUS MULT DIV ID NUMBER

%%
program:
    while_statement
  ;

while_statement:
```

```
    WHILE LPAREN condition RPAREN LBRACE statements RBRACE
     { printf("Valid WHILE loop syntax\n"); }
  ;

condition:
    ID LT NUMBER
  | ID GT NUMBER
  | ID LE NUMBER
  | ID GE NUMBER
  | ID EQ NUMBER
  | ID NE NUMBER
  | ID LT ID
  | ID GT ID
  | ID LE ID
  | ID GE ID
  | ID EQ ID
  | ID NE ID
  | NUMBER LT NUMBER
  | NUMBER GT NUMBER
  | NUMBER LE NUMBER
  | NUMBER GE NUMBER
  | NUMBER EQ NUMBER
  | NUMBER NE NUMBER
  ;

statements:
    statement
  | statements statement
  ;

statement:
    assignment_statement
  | expression_statement
  | increment_statement
  | function_call
  ;

assignment_statement:
    ID ASSIGN expression SEMICOLON
  ;

expression_statement:
```

```
    expression SEMICOLON
  ;

increment_statement:
    ID INC SEMICOLON
  | ID DEC SEMICOLON
  ;

function_call:
    ID LPAREN arguments RPAREN SEMICOLON
  ;

arguments:
    /* empty */
  | expression
  | arguments ',' expression
  ;

expression:
    ID
  | NUMBER
  | ID PLUS NUMBER
  | ID MINUS NUMBER
  | ID MULT NUMBER
  | ID DIV NUMBER
  | NUMBER PLUS NUMBER
  | NUMBER MINUS NUMBER
  | NUMBER MULT NUMBER
  | NUMBER DIV NUMBER
  | ID PLUS ID
  | ID MINUS ID
  | ID MULT ID
  | ID DIV ID
  ;

%%

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

int main(void) {
```

```
    printf("Enter a WHILE loop statement:\n");
    yyparse();
    return 0;
}
```

OUTPUT

```
PS C:\Users\Soham\Documents\SEM7\Sem7 codes> ./while_loop_checker
Enter a WHILE loop statement:
while(i<10) {i++;}
Valid WHILE loop syntax
while(i>1)
Syntax error: syntax error
```

Q3:

Code:

Lex file:

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

%%

"switch"        { return SWITCH; }
"case"          { return CASE; }
"default"       { return DEFAULT; }
"break"         { return BREAK; }
"int"           { return INT; }
"("             { return LPAREN; }
")"             { return RPAREN; }
"{"             { return LBRACE; }
"}"             { return RBRACE; }
":"             { return COLON; }
";"             { return SEMICOLON; }
"="             { return ASSIGN; }
"+"             { return PLUS; }
"-"             { return MINUS; }
"*"             { return MULT; }
"/"             { return DIV; }
","             { return COMMA; }
[0-9]+           { yylval = atoi(yytext); return NUMBER; }
[a-zA-Z_][a-zA-Z0-9_]* { return ID; }
[ \t\n\r]+      ;
.               { return yytext[0]; }

%%
```

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

Y.file:

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

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

%token SWITCH CASE DEFAULT BREAK INT LPAREN RPAREN LBRACE RBRACE
COLON SEMICOLON ASSIGN PLUS MINUS MULT DIV COMMA ID NUMBER

%%
program:
    switch_statement
  ;

switch_statement:
    SWITCH LPAREN expression RPAREN LBRACE case_list default_clause_opt
RBRACE
     { printf("Valid SWITCH-CASE syntax\n"); }
  ;

case_list:
    /* empty */
  | case_list case_clause
  ;

case_clause:
    CASE case_value COLON statements break_opt
  ;
```

```
case_value:
     NUMBER
   | ID
   ;

default_clause_opt:
     /* empty */
   | DEFAULT COLON statements
   ;

break_opt:
     /* empty */
   | BREAK SEMICOLON
   ;

statements:
     /* empty */
   | statements statement
   ;

statement:
     declaration
   | assignment
   | expression_statement
   | function_call
   ;

declaration:
     INT ID initializer_opt SEMICOLON
   ;

initializer_opt:
     /* empty */
```

```
    | ASSIGN expression
    ;

assignment:
     ID ASSIGN expression SEMICOLON
    ;

expression_statement:
     expression SEMICOLON
    ;

function_call:
     ID LPAREN arg_list_opt RPAREN SEMICOLON
    ;

arg_list_opt:
     /* empty */
    | arg_list
    ;

arg_list:
     expression
    | arg_list COMMA expression
    ;

expression:
     ID
    | NUMBER
    | expression PLUS expression
    | expression MINUS expression
    | expression MULT expression
    | expression DIV expression
    ;
```

```
%%

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

int main(void) {
   printf("Enter a SWITCH-CASE statement:\n");
   yyparse();
   return 0;
}
```

Output:

```
PS C:\Users\Soham\Documents\SEM7\Sem7 codes> ./switch_case
Enter a SWITCH-CASE statement:
switch(x){ case 1: break; default: x=0;}
Valid SWITCH-CASE syntax
switch(x){ case 1: default:}
Syntax error: syntax error
```

POST LAB Question:

Q. 1 YACC program for checking syntax for an If then.

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

%%

"if"            { return IF; }
"int"           { return INT; }
"("             { return LPAREN; }
")"             { return RPAREN; }
"{"             { return LBRACE; }
"}"             { return RBRACE; }
";"             { return SEMICOLON; }
"="             { return ASSIGN; }
"<="            { return LE; }
">="            { return GE; }
"=="            { return EQ; }
"!="            { return NE; }
"<"             { return LT; }
```

```
">"             { return GT; }
"&&"            { return AND; }
"||"            { return OR; }
"!"             { return NOT; }
"+"             { return PLUS; }
"-"             { return MINUS; }
"*"             { return MULT; }
"/"             { return DIV; }
[0-9]+          { yylval = atoi(yytext); return NUMBER; }
[a-zA-Z_][a-zA-Z0-9_]* { return ID; }
[ \t\n\r]+      ;
.               { return yytext[0]; }

%%

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

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

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

%token IF INT LPAREN RPAREN LBRACE RBRACE SEMICOLON ASSIGN LT GT LE GE EQ NE AND OR
NOT PLUS MINUS MULT DIV ID NUMBER

%%
program:
    if_statement
  ;

if_statement:
    IF LPAREN condition RPAREN LBRACE statements RBRACE
     { printf("Valid IF-THEN syntax\n"); }
  ;

condition:
    expression
  | condition AND condition
  | condition OR condition
  | NOT condition
  | LPAREN condition RPAREN
```

```
    ;

statements:
     statement
    | statements statement
    ;

statement:
     declaration
    | assignment
    | expression_statement
    | function_call
    ;

declaration:
     INT ID initializer_opt SEMICOLON
    ;

initializer_opt:
     /* empty */
    | ASSIGN expression
    ;

assignment:
     ID ASSIGN expression SEMICOLON
    ;

expression_statement:
     expression SEMICOLON
    ;

function_call:
     ID LPAREN arg_list_opt RPAREN SEMICOLON
    ;

arg_list_opt:
     /* empty */
    | arg_list
    ;

arg_list:
     expression
```

```
    | arg_list ',' expression
    ;

expression:
     ID
    | NUMBER
    | expression LT expression
    | expression GT expression
    | expression LE expression
    | expression GE expression
    | expression EQ expression
    | expression NE expression
    | expression PLUS expression
    | expression MINUS expression
    | expression MULT expression
    | expression DIV expression
    | LPAREN expression RPAREN
    ;

%%

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

int main(void) {
    printf("Enter an IF-THEN statement:\n");
    yyparse();
    return 0;
}
```

```
PS C:\Users\Soham\Documents\SEM7\Sem7 codes> ./if_then
Enter an IF-THEN statement:
if(x > 5) { int a = 5; x = x + 1; }
Valid IF-THEN syntax
if(x >) { a =  }
Syntax error: syntax error
```

Q2. **YACC program for checking syntax for an If then else.**

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

```
%%

"if"            { return IF; }
"else"          { return ELSE; }
"int"           { return INT; }
"("             { return LPAREN; }
")"             { return RPAREN; }
"{"             { return LBRACE; }
"}"             { return RBRACE; }
";"             { return SEMICOLON; }
"="             { return ASSIGN; }
"<="            { return LE; }
">="            { return GE; }
"=="            { return EQ; }
"!="            { return NE; }
"<"             { return LT; }
">"             { return GT; }
"&&"            { return AND; }
"||"            { return OR; }
"!"             { return NOT; }
"+"             { return PLUS; }
"-"             { return MINUS; }
"*"             { return MULT; }
"/"             { return DIV; }
[0-9]+          { yylval = atoi(yytext); return NUMBER; }
[a-zA-Z_][a-zA-Z0-9_]* { return ID; }
[ \t\n\r]+      ;
.               { return yytext[0]; }

%%

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

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

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

%token IF ELSE INT LPAREN RPAREN LBRACE RBRACE SEMICOLON ASSIGN LT GT LE GE EQ
NE AND OR NOT PLUS MINUS MULT DIV ID NUMBER
```

```
%%
program:
    if_else_statement
  ;

if_else_statement:
    IF LPAREN condition RPAREN LBRACE statements RBRACE ELSE LBRACE statements
RBRACE
     { printf("Valid IF-THEN-ELSE syntax\n"); }
  ;

condition:
    expression
  | condition AND condition
  | condition OR condition
  | NOT condition
  | LPAREN condition RPAREN
  ;

statements:
    statement
  | statements statement
  ;

statement:
    declaration
  | assignment
  | expression_statement
  | function_call
  ;

declaration:
    INT ID initializer_opt SEMICOLON
  ;

initializer_opt:
    /* empty */
  | ASSIGN expression
  ;
```

```
assignment:
    ID ASSIGN expression SEMICOLON
  ;

expression_statement:
    expression SEMICOLON
  ;

function_call:
    ID LPAREN arg_list_opt RPAREN SEMICOLON
  ;

arg_list_opt:
    /* empty */
  | arg_list
  ;

arg_list:
    expression
  | arg_list ',' expression
  ;

expression:
    ID
  | NUMBER
  | expression LT expression
  | expression GT expression
  | expression LE expression
  | expression GE expression
  | expression EQ expression
  | expression NE expression
  | expression PLUS expression
  | expression MINUS expression
  | expression MULT expression
  | expression DIV expression
  | LPAREN expression RPAREN
  ;
```

```
%%

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

int main(void) {
   printf("Enter an IF-THEN-ELSE statement:\n");
   yyparse();
   return 0;
}
```

```
Enter an IF-THEN-ELSE statement:
if(x > 0) { x = x + 1; } else { x = 0; }
Valid IF-THEN-ELSE syntax
if(x >) { x = x + 1 } else
Syntax error: syntax error
```

Q3. YACC program for checking syntax for custom functions.

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

%%

"int"           { return INT; }
"void"          { return VOID; }
"return"        { return RETURN; }
"("             { return LPAREN; }
")"             { return RPAREN; }
"{"             { return LBRACE; }
"}"             { return RBRACE; }
";"             { return SEMICOLON; }
","             { return COMMA; }
"="             { return ASSIGN; }
"<="            { return LE; }
">="            { return GE; }
"=="            { return EQ; }
"!="            { return NE; }
"<"             { return LT; }
">"             { return GT; }
```

```
"&&"            { return AND; }
"||"            { return OR; }
"!"             { return NOT; }
"+"             { return PLUS; }
"-"             { return MINUS; }
"*"             { return MULT; }
"/"             { return DIV; }
[0-9]+          { yylval = atoi(yytext); return NUMBER; }
[a-zA-Z_][a-zA-Z0-9_]* { return ID; }
[ \t\n\r]+      ;
.               { return yytext[0]; }

%%

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

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

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

%token INT VOID RETURN LPAREN RPAREN LBRACE RBRACE SEMICOLON COMMA ASSIGN LT GT LE GE EQ NE AND OR NOT PLUS MINUS MULT DIV ID NUMBER

%%
program:
function_definition
;

function_definition:
return_type ID LPAREN parameter_list_opt RPAREN LBRACE function_body RBRACE
{ printf("Valid CUSTOM FUNCTION syntax\n"); }
;

return_type:
INT
| VOID
;

parameter_list_opt:
/* empty */
| parameter_list
;

parameter_list:
parameter
| parameter_list COMMA parameter
;

parameter:
INT ID
;

function_body:
statements return_statement_opt
;

statements:
/* empty */
| statements statement
;

statement:
declaration
| assignment
| expression_statement
| function_call
| if_statement
| while_statement
;

declaration:
INT ID initializer_opt SEMICOLON
;

initializer_opt:
/* empty */
| ASSIGN expression

;

assignment:
ID ASSIGN expression SEMICOLON
;

expression_statement:
expression SEMICOLON
;

function_call:
ID LPAREN arg_list_opt RPAREN SEMICOLON
;

arg_list_opt:
/* empty */
| arg_list
;

arg_list:
expression
| arg_list COMMA expression
;

if_statement:
IF LPAREN condition RPAREN LBRACE statements RBRACE
;

while_statement:
WHILE LPAREN condition RPAREN LBRACE statements RBRACE
;

return_statement_opt:
/* empty */
| RETURN expression_opt SEMICOLON
;

expression_opt:
/* empty */

```
    | expression
    ;

condition:
      expression
    | condition AND condition
    | condition OR condition
    | NOT condition
    | LPAREN condition RPAREN
    ;

expression:
      ID
    | NUMBER
    | expression LT expression
    | expression GT expression
    | expression LE expression
    | expression GE expression
    | expression EQ expression
    | expression NE expression
    | expression PLUS expression
    | expression MINUS expression
    | expression MULT expression
    | expression DIV expression
    | LPAREN expression RPAREN
    ;

%%

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

int main(void) {
    printf("Enter a CUSTOM FUNCTION definition:\n");
    yyparse();
    return 0;
}
```

```
PS C:\Users\Soham\Documents\SEM7\Sem7 codes> ./custom_functions
Enter a CUSTOM FUNCTION definition:
int add(int a, int b) { int result = a + b; return result; }
Valid CUSTOM FUNCTION syntax
int add(int a, int b int result = a + b return result
Syntax error: syntax error
```