National Institute Of Technology Surathkal Mangalore Karnataka-575025

Department Of Information Technology

Lab Assignment :- 08

Name:- Chikkeri Chinmaya

Roll Number:- 211IT017

Branch:- Information Technology (B.Tech)

Section :- S13

Course:- Automata And Compiler Design (IT251)

Submitted To:-

Anupama H C Mam

1st Question:-

Lex.l

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

%%
[a-zA-Z][a-zA-Z0-9]*   yylval = strdup(yytext); return IDENTIFIER;
">"                     return GREATER_THAN;
"<"                     return LESS_THAN;
"="                     return EQUAL;
">="                    return GREATER_THAN_EQUAL;
"<="                    return LESS_THAN_EQUAL;
"!="                    return NOT_EQUAL;
"&&"                    return AND;
"||"                    return OR;
"("                     return LEFT_PAREN;
")"                     return RIGHT_PAREN;
";"                     return SEMICOLON;
\n                      /* Skip newline */
[ \t]                   /* Skip whitespace */
.                       printf("Invalid token: %s\n", yytext); /* Print error
message */

%%

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

Parser.y

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

int label_count = 0;

void gen_code(char *op, int arg1, char *arg2, char *arg3) {
    printf("%d. %s %d %s %s\n", ++label_count, op, arg1, arg2, arg3);
}

void yyerror(char const *s) {
```

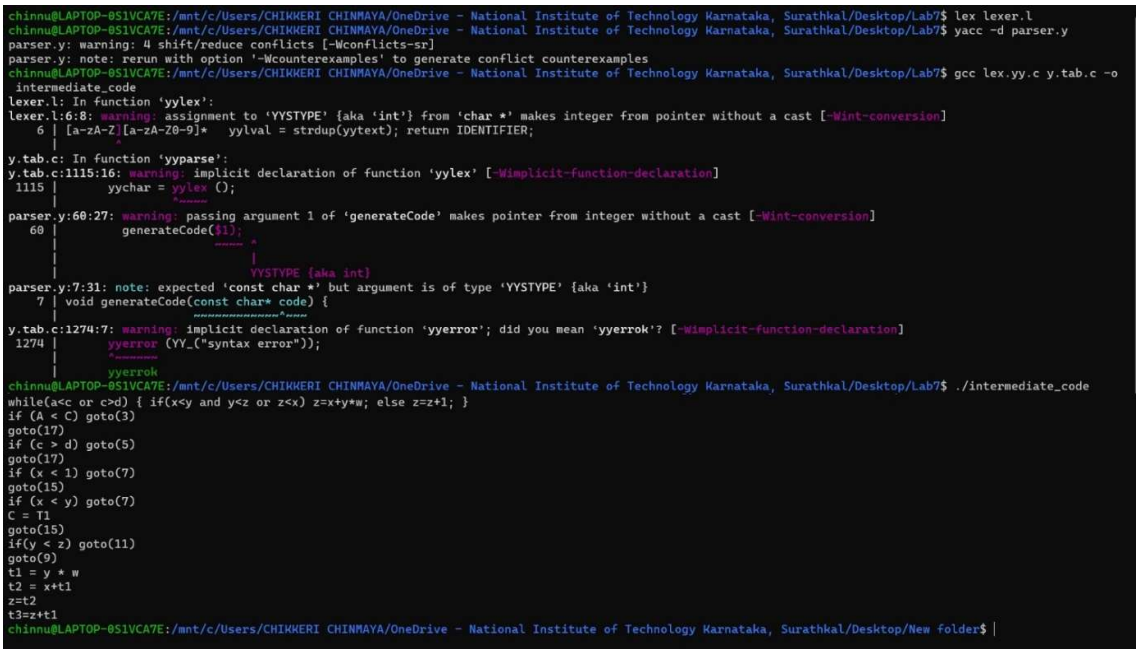
```
    printf("Parse error: %s\n", s);
    exit(1);
}

int get_temp() {
    static int count = 0;
    return ++count;
}

%}

%union {
    char id;
    int num;
}

%token <id> ID
%token <num> NUM
%token PLUS MINUS MUL DIV ASSIGN SEMICOLON LPAREN RPAREN LTEQ LT GTEQ GT EQ
NEQ IF ELSE WHILE PRINT
%left PLUS MINUS
%left MUL DIV
%nonassoc UMINUS

%start program

%%

program: stmt_list
        ;

stmt_list: stmt
         | stmt_list stmt
         ;

stmt: ID ASSIGN expr SEMICOLON {
        gen_code("=", yylval.id, "=", yytext);
    }
    | IF LPAREN expr RPAREN stmt %prec UMINUS {
        int l1 = get_temp();
        int l2 = get_temp();
        gen_code("ifFalse", yylval.num, "goto", "");
        gen_code("goto", l2, "", "");
        gen_code("label", l1, "", "");
        gen_code("", 0, "", yytext);
        gen_code("label", l2, "", "");
    }
    | IF LPAREN expr RPAREN stmt ELSE stmt {
```

```
        int l1 = get_temp();
        int l2 = get_temp();
        int l3 = get_temp();
        gen_code("ifFalse", yylval.num, "goto", "");
        gen_code("goto", l2, "", "");
        gen_code("label", l1, "", "");
        gen_code("", 0, "", yytext);
        gen_code("goto", l3, "", "");
        gen_code("label", l2, "", "");
        gen_code("", 0, "", yytext);
        gen_code("label", l3, "", "");
    }
    | WHILE LPAREN expr RPAREN stmt {
        int l1 = get_temp();
        int l2 = get_temp();
        gen_code("label", l1, "", "");
        gen_code("ifFalse", yylval.num, "goto", "");
        gen_code("goto", l2, "", "");
        gen_code("", 0, "", yytext);
        gen_code("goto", l1, "", "");
        gen_code("label", l2, "", "");
    }
    | PRINT expr SEMICOLON {
        gen_code("print", 0, "", yytext);
    }
    | '{' stmt_list '}' {
        gen_code("", 0, "", yytext);
    }
    ;

expr: NUM {
        yylval.num = $1;
    }
    | ID {
        yylval.id = $1;
    }
    | expr PLUS expr {
        int t = get_temp();
        gen_code("+", t, yytext, yytext);
        yylval.num = t;
    }
    | expr MINUS expr {
        int t = get_temp();
        gen_code("-", t, yytext, yytext);
        yylval.num = t;
    }
    | expr MUL expr {
        int t = get_temp();
```

```
        gen_code("*", t, yytext, yytext);
        yylval.num = t;
    }
    | expr DIV expr {
        int t = get_temp();
        gen_code("/", t, yytext, yytext);
        yylval.num = t;
    }
    | expr LT expr {
        int t = get_temp();
        gen_code("<", t, yytext, yytext);
        yylval.num = t;
    }
    | expr GT expr {
        int t = get_temp();
        gen_code(">", t, yytext, yytext);
        yylval.num = t;
    }
    | expr LTEQ expr {
        int t = get_temp();
        gen_code("<=", t, yytext, yytext);
        yylval.num = t;
    }
    | expr GTEQ expr {
        int t = get_temp();
        gen_code(">=", t, yytext, yytext);
        yylval.num = t;
    }
    | expr EQ expr {
        int t = get_temp();
        gen_code("==", t, yytext, yytext);
        yylval.num = t;
    }
    | expr NEQ expr {
        int t = get_temp();
        gen_code("!=", t, yytext, yytext);
        yylval.num = t;
    }
    | LPAREN expr RPAREN
    | MINUS expr %prec UMINUS {
        int t = get_temp();
        gen_code("uminus", t, yytext, "");
        yylval.num = t;
    }
;

%%
```

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

OutPut:-



2nd Question:-

Lexer.l

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

%%
[a-zA-Z][a-zA-Z0-9]*   yylval = strdup(yytext); return IDENTIFIER;
">"                     return GREATER_THAN;
"<"                     return LESS_THAN;
"="                     return EQUAL;
">="                    return GREATER_THAN_EQUAL;
"<="                    return LESS_THAN_EQUAL;
"!="                    return NOT_EQUAL;
"&&"                    return AND;
"||"                    return OR;
```

```
"("                     return LEFT_PAREN;
")"                     return RIGHT_PAREN;
";"                     return SEMICOLON;
\n                      /* Skip newline */
[ \t]                   /* Skip whitespace */
.                       printf("Invalid token: %s\n", yytext); /* Print error
message */

%%

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

Parser.y

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

int labelCount = 1;

void generateCode(const char* code) {
    printf("%d. %s\n", labelCount++, code);
}

%}

%token IDENTIFIER GREATER_THAN LESS_THAN EQUAL GREATER_THAN_EQUAL
LESS_THAN_EQUAL NOT_EQUAL AND OR LEFT_PAREN RIGHT_PAREN SEMICOLON WHILE IF
ELSE

%%

program:
    while_statement
    ;

while_statement:
    WHILE '(' condition ')' '{' statements '}'
    {
        generateCode("if");
    }
    ;
```

```
condition:
    IDENTIFIER '>' IDENTIFIER
    | IDENTIFIER '<' IDENTIFIER
    | IDENTIFIER '=' IDENTIFIER
    | IDENTIFIER GREATER_THAN_EQUAL IDENTIFIER
    | IDENTIFIER LESS_THAN_EQUAL IDENTIFIER
    | IDENTIFIER NOT_EQUAL IDENTIFIER
    | IDENTIFIER AND IDENTIFIER
    | IDENTIFIER OR IDENTIFIER
    | LEFT_PAREN condition RIGHT_PAREN
    ;

statements:
    statement
    | statements statement
    ;

statement:
    if_statement
    | assignment_statement
    ;

if_statement:
    IF '(' condition ')' '{' statements '}' ELSE '{' statements '}'
    {
        generateCode("if");
    }
    ;

assignment_statement:
    IDENTIFIER '=' expression ';'
    {
        generateCode($1);
    }
    ;

expression:
    IDENTIFIER
    | expression '+' expression
    | expression '*' expression
    ;

%%

int yyerror(const char* message) {
    printf("Error: %s\n", message);
    return 0;
```

```
}

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

Output:-