TAGORE ENGINEERING COLLEGE

RATHINAMANGALAM, CHENNAI-127.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS3501-COMPILER DESIGN LABORATORY

Name	**:**	________________________
Reg. No	**:**	________________________
Branch	**:**	________________________
Year	**:**	________________________
Semester	**:**	________________________

Anna University::
Chennai Regulation 2021

TAGORE ENGINEERING COLLEGE

RATHINAMANGALAM, CHENNAI-600127

LABORATORY RECORD

UNIVERSITY REGISTER NO.

Certified that this is the bonafide record of work done by
Mr. /Ms.____of______________
Department in the____laboratory
and submitted for the University Practical Examination conducted on ________________
at TAGORE ENGINEERING COLLEGE, CHENNAI-127.

Record Mark: _______

Lab In-charge **Principal** **Head of the Department**

External Examiner **Internal Examiner**

VISION

To create an environment which is conducive to produce competent Computer Science Engineers through quality education and research-oriented education and equip them for the needs of the industry and society.

MISSION

The Department strives to contribute to the expansion of knowledge in the discipline of Computer Science and Engineering by

- Adopting an efficient teaching learning process in concurrence with increasing industrial demands.
- Ensuring technical proficiency, facilitating to pursue higher studies and carry out Research & Development activities.
- Developing problem solving and analytical skills with deep knowledge in thorough understanding of basic sciences.
- and Computer Science Engineering.
- Infusing managerial and entrepreneurship skills to become ethical, socially responsible, and competitive professionals.

PROGRAM SPECIFIC OBJECTIVES (PSOs)

1. Exhibit design and programming skills to build and automate business solutions using cutting edge technologies

2. Strong theoretical foundation leading to excellence and excitement towards research, to provide elegant solutions to complex problems.

3. Ability to work effectively with various engineering fields as a team to design, build and develop system applications.

PROGRAM OUTCOMES POs

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one‘s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

COURSE OUTCOMES

CO1: Understand the techniques in different phases of a compiler.
CO2: Design a lexical analyser for a sample language and learn to use the LEX tool.
CO3: Apply different parsing algorithms to develop a parser and learn to use YACC tool.
CO4: Understand semantics rules (SDT), intermediate code generation and run-time environment.
CO5: Implement code generation and apply code optimization techniques.

TABLE OF CONTENTS

PROGRAM:

```
 %{
#include <stdio.h>
#include <string.h>
#define MAX_SYMBOLS 100
typedef struct {
   char name[50];
   int line;
} SymbolInfo;
SymbolInfo symbolTable[MAX_SYMBOLS];
int symbolCount = 0;
void addSymbol(const char* name, int line) {
   if (symbolCount < MAX_SYMBOLS && strcmp(name, "stdio.h") != 0) {
      strcpy(symbolTable[symbolCount].name, name);
      symbolTable[symbolCount].line = line;
      symbolCount++;}
}
%}
%option noyywrap
%x COMMENT
%{
   int line_no = 1;
%}
%{
   char* keywords[] = {"int", "void", "main", "char", "if", "for", "while", "else", "printf",
"scanf", "FILE", "include","stdioh","conio.h"};
   int numKeywords = sizeof(keywords) / sizeof(keywords[0]);
%}
%%
[a-zA-Z_][a-zA-Z0-9_]* {
   int isKeyword = 0;
   for (int i = 0; i < numKeywords; i++) {
      if (strcmp(yytext, keywords[i]) == 0) {
         isKeyword = 1;
         break;
      }
   }
   if (isKeyword) {
      printf("Keyword: %s\n", yytext);
   } else {
      printf("Identifier: %s\n", yytext);
```

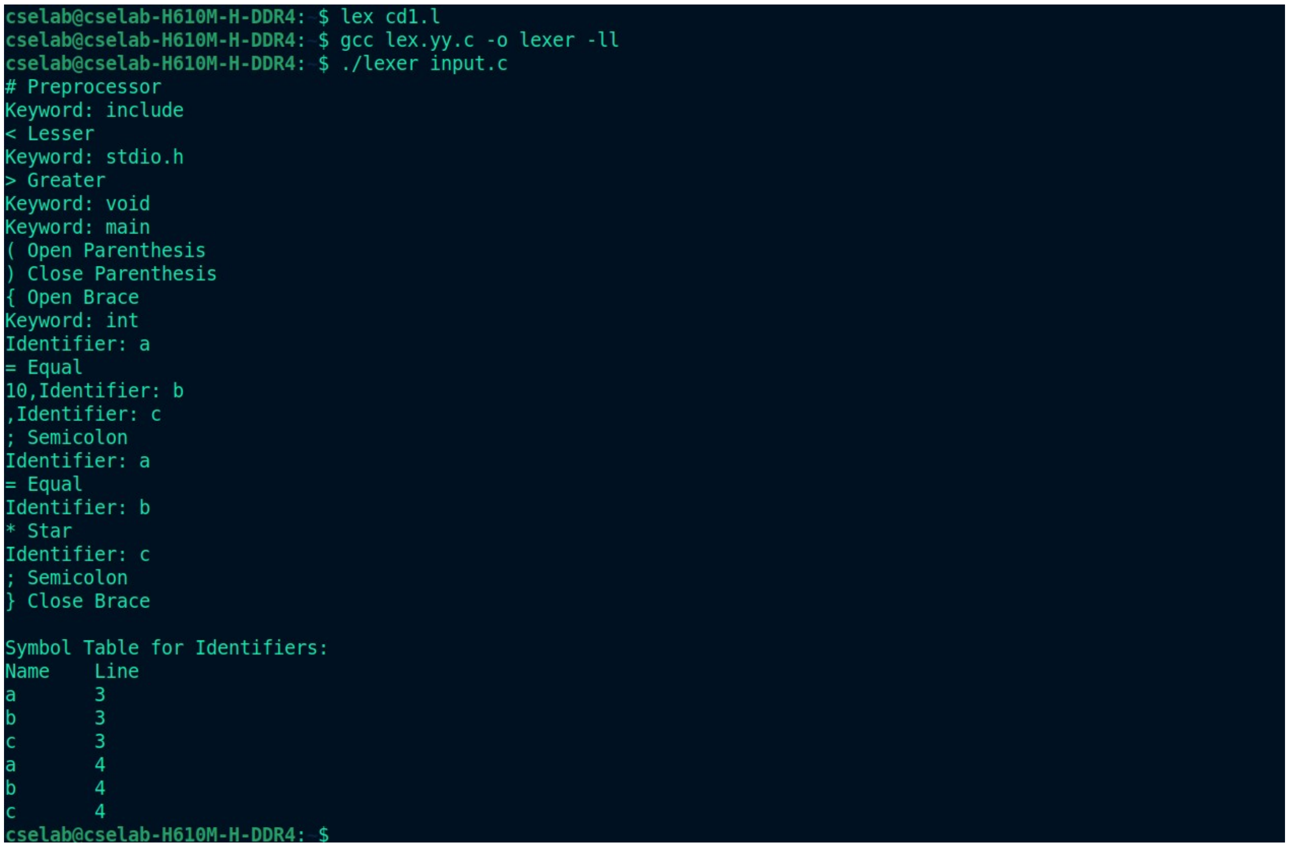
```
        addSymbol(yytext, line_no);
    }
}
"stdio.h" { printf("Keyword: stdio.h\n"); addSymbol(yytext, line_no); }
\( {    printf("( Open Parenthesis\n");}
\) {    printf(") Close Parenthesis\n");}
\{ {    printf("{ Open Brace\n");}
\} {    printf("} Close Brace\n");}
\< {    printf("< Lesser\n");}
\> {    printf("> Greater\n");}
\" {    printf("\" Double Quote\n");}
\' {    printf("\' Single Quote\n");}
\: {    printf(": Colon\n");}
\; {    printf("; Semicolon\n");}
\# {    printf("# Preprocessor\n");}
\= {    printf("= Equal\n");}
\== {    printf("== Assignment\n");}
\% {    printf("%% Percentage\n");}
\^ {    printf("^ Bitwise\n");}
\& {    printf("& Reference\n");}
\* {    printf("* Star\n");}
\+ {    printf("+ Addition\n");}
\- {    printf("- Subtraction\n");}
\\ {    printf("\\ Backslash\n");}
\/ {    printf("/ Slash\n");}
\n {    line_no++;}
[ \t] ;
\/\/[^\n]* ;
\/\*[^*]*\*+([^/*][^*]*\*+)*\/ ;
%%
int main(int argc, char* argv[]) {
    if (argc < 2) {
        printf("Usage: %s <input_file>\n", argv[0]);
        return 1;
    }
    FILE* input_file = fopen(argv[1], "r");
    if (!input_file) {
        perror("Error opening input file");
        return 1;
    }
    yyin = input_file;
```

```
        yylex();
        printf("\nSymbol Table for Identifiers:\n");
        printf("Name\tLine\n");
        for (int i = 0; i < symbolCount; i++) {
            printf("%s\t%d\n", symbolTable[i].name, symbolTable[i].line);
        }
        fclose(input_file);
        return 0;
    }
```

Input.c

```
#include <stdio.h>
void main() {
   int a = 10, b, c;
   a = b * c;
}
```

OUTPUT:



```
cselab@cselab-H610M-H-DDR4: $ lex cd1.l
cselab@cselab-H610M-H-DDR4: $ gcc lex.yy.c -o lexer -ll
cselab@cselab-H610M-H-DDR4: $ ./lexer input.c
# Preprocessor
Keyword: include
< Lesser
Keyword: stdio.h
> Greater
Keyword: void
Keyword: main
( Open Parenthesis
) Close Parenthesis
{ Open Brace
Keyword: int
Identifier: a
= Equal
10,Identifier: b
,Identifier: c
; Semicolon
Identifier: a
= Equal
Identifier: b
* Star
Identifier: c
; Semicolon
} Close Brace

Symbol Table for Identifiers:
Name    Line
a       3
b       3
c       3
a       4
b       4
c       4
cselab@cselab-H610M-H-DDR4: $
```

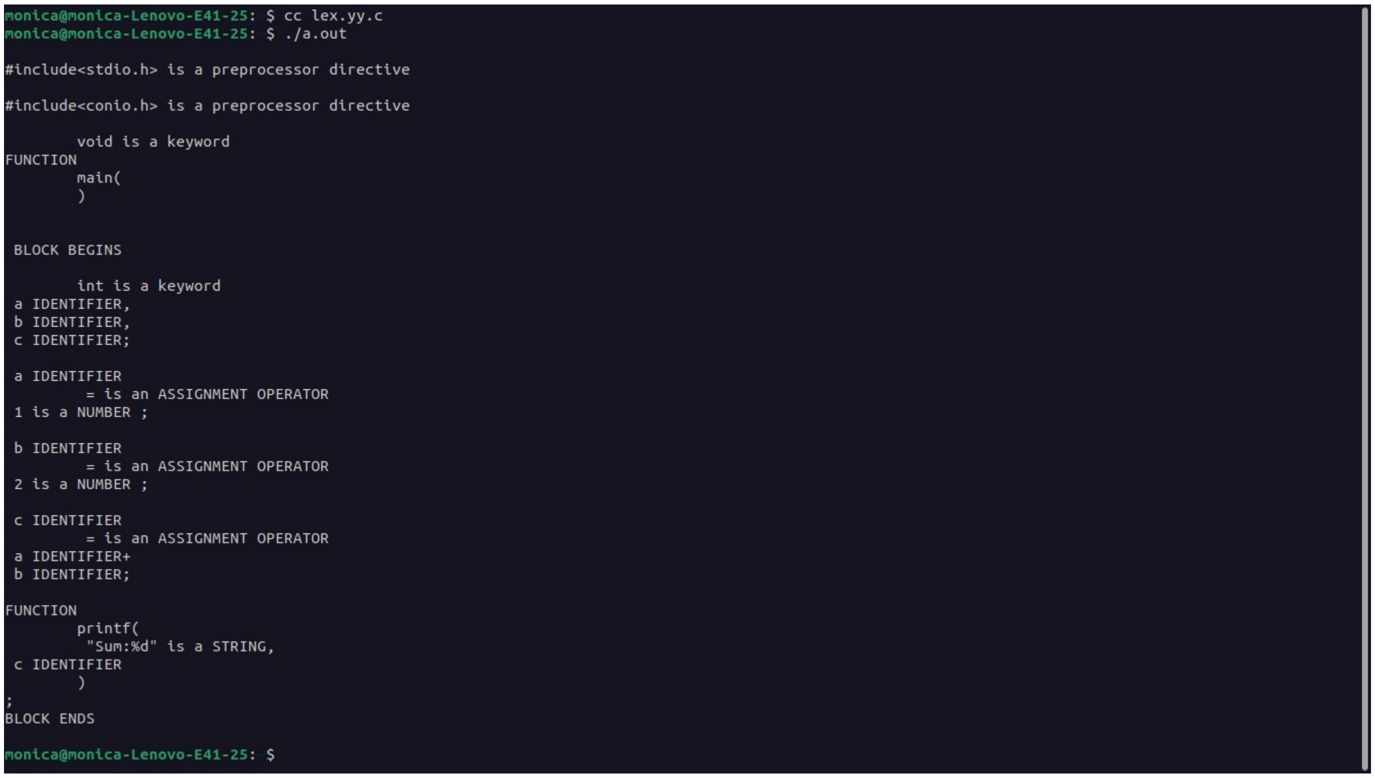
PROGRAM:

```
%{
int COMMENT = 0;
%}
identifier [a-zA-Z][a-zA-Z0-9]*
%%
#.* {printf("\n%s is a preprocessor directive", yytext);}
int |
float |
char |
double |
while |
for |
struct |
typedef |
do |
if |
break |
continue |
void |
switch |
return |
else |
goto {printf("\n\t%s is a keyword", yytext);}
"/*" {COMMENT = 1;} {printf("\n\t %s is a COMMENT", yytext);}
{identifier}\( {if (!COMMENT) printf("\nFUNCTION \n\t%s", yytext);}
\{ {if (!COMMENT) printf("\n BLOCK BEGINS");}
\} {if (!COMMENT) printf("BLOCK ENDS ");}
{identifier}(\[[0-9]*\])? {if (!COMMENT) printf("\n %s IDENTIFIER", yytext);}
\".*\" {if (!COMMENT) printf("\n\t %s is a STRING", yytext);}
[0-9]+ {if (!COMMENT) printf("\n %s is a NUMBER ",
yytext);}
\)(\:)? {if (!COMMENT) printf("\n\t"); ECHO; printf("\n");}
\( ECHO;
= {if (!COMMENT) printf("\n\t %s is an ASSIGNMENT OPERATOR", yytext);}
\<= |
\>= |
\< |
== |
\> {if (!COMMENT) printf("\n\t%s is a RELATIONAL OPERATOR", yytext);}
%%
int main(int argc, char **argv) {
   FILE *file;
   file = fopen("var.c", "r");
   if (!file) {
      printf("Could not open the file");
      exit(0);
   }
   yyin = file;
   yylex();
   printf("\n");
   return 0;
}
int yywrap() {
   return 1;
}
```

var.c

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
a=1;
b=2;
c=a+b; printf("Sum:
%d",c);
}
```

OUTPUT:



```
monica@monica-Lenovo-E41-25: $ cc lex.yy.c
monica@monica-Lenovo-E41-25: $ ./a.out

#include<stdio.h> is a preprocessor directive

#include<conio.h> is a preprocessor directive

        void is a keyword
FUNCTION
        main(
        )


 BLOCK BEGINS

        int is a keyword
 a IDENTIFIER,
 b IDENTIFIER,
 c IDENTIFIER;

 a IDENTIFIER
         = is an ASSIGNMENT OPERATOR
 1 is a NUMBER ;

 b IDENTIFIER
         = is an ASSIGNMENT OPERATOR
 2 is a NUMBER ;

 c IDENTIFIER
         = is an ASSIGNMENT OPERATOR
 a IDENTIFIER+
 b IDENTIFIER;

FUNCTION
        printf(
         "Sum:%d" is a STRING,
 c IDENTIFIER
        )
;
BLOCK ENDS

monica@monica-Lenovo-E41-25: $
```

PROGRAM:

LEX PART : arithmetic.l

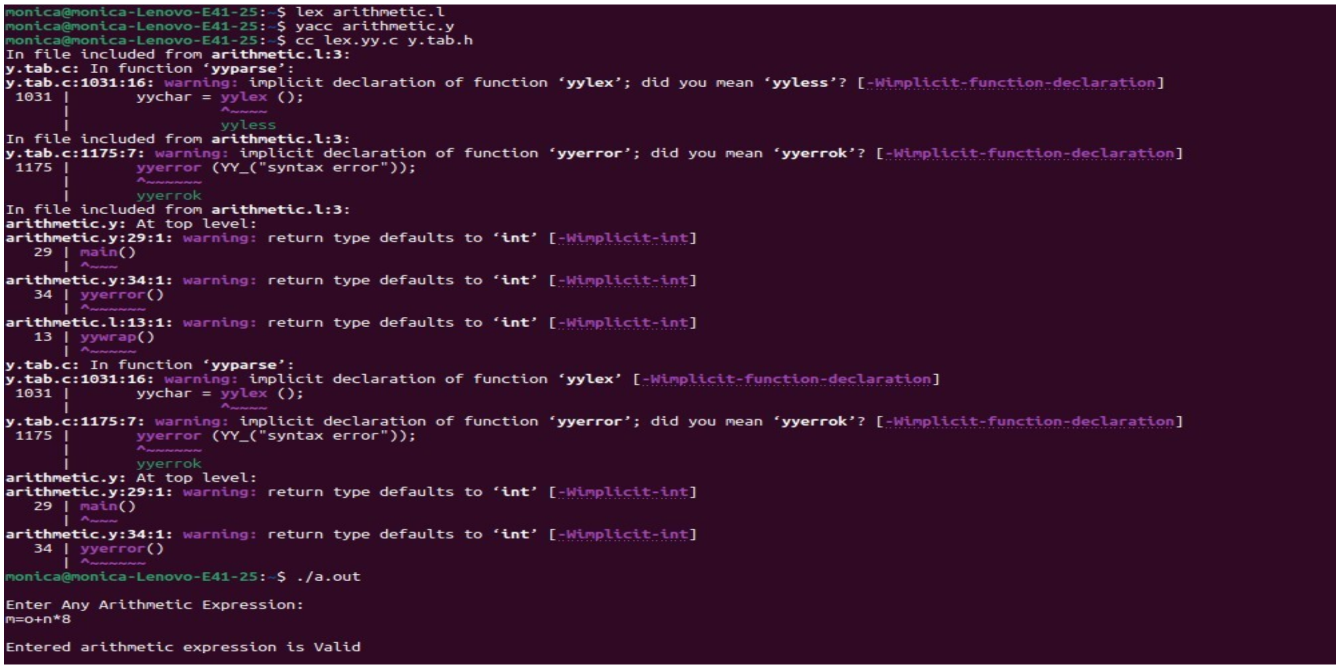
```
%{
#include<stdio.h>
#include "y.tab.h"
%}
%%
[a-zA-Z] {return
VARIABLE;} [0-9] {return
NUMBER;}
[\t] ;
[\n] {return 0;}
. {return yytext[0];}
%%
yywrap()
{}
```

YACC PART : arithmetic.y

```
%{
   #include<stdio.h>
%}
%token NUMBER
%token VARIABLE
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
S: VARIABLE'='E {
      printf("\nEntered arithmetic expression is Valid\n\n");
      return 0;
     }
E:E'+'E
 |E'-'E
 |E'*'E
 |E'/'E
 |E'%'E
 |'('E')'
 | NUMBER
 | VARIABLE
;
%%
main()
{
  printf("\nEnter Any Arithmetic Expression:\n");
  yyparse();
}
yyerror()
{
  printf("\nEntered arithmetic expression is Invalid\n\n");
}
```

OUTPUT:

```
$:lex arithmetic.l
$:yacc arithmetic.y
$:cc lex.yy.c y.tab.h
$:./a.out
```



```
monica@monica-Lenovo-E41-25:~$ lex arithmetic.l
monica@monica-Lenovo-E41-25:~$ yacc arithmetic.y
monica@monica-Lenovo-E41-25:~$ cc lex.yy.c y.tab.h
In file included from arithmetic.l:3:
y.tab.c: In function ‘yyparse’:
y.tab.c:1031:16: warning: implicit declaration of function ‘yylex’; did you mean ‘yyless’? [-Wimplicit-function-declaration]
 1031 |       yychar = yylex ();
      |                ^~~~~
      |                yyless
In file included from arithmetic.l:3:
y.tab.c:1175:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
 1175 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
In file included from arithmetic.l:3:
arithmetic.y: At top level:
arithmetic.y:29:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   29 | main()
      | ^~~~
arithmetic.y:34:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   34 | yyerror()
      | ^~~~~~~
arithmetic.l:13:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   13 | yywrap()
      | ^~~~~~
y.tab.c: In function ‘yyparse’:
y.tab.c:1031:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
 1031 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1175:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
 1175 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
arithmetic.y: At top level:
arithmetic.y:29:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   29 | main()
      | ^~~~
arithmetic.y:34:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   34 | yyerror()
      | ^~~~~~~
monica@monica-Lenovo-E41-25:~$ ./a.out

Enter Any Arithmetic Expression:
m=o+n*8

Entered arithmetic expression is Valid
```

PROGRAM:

LEX PART : variable.l

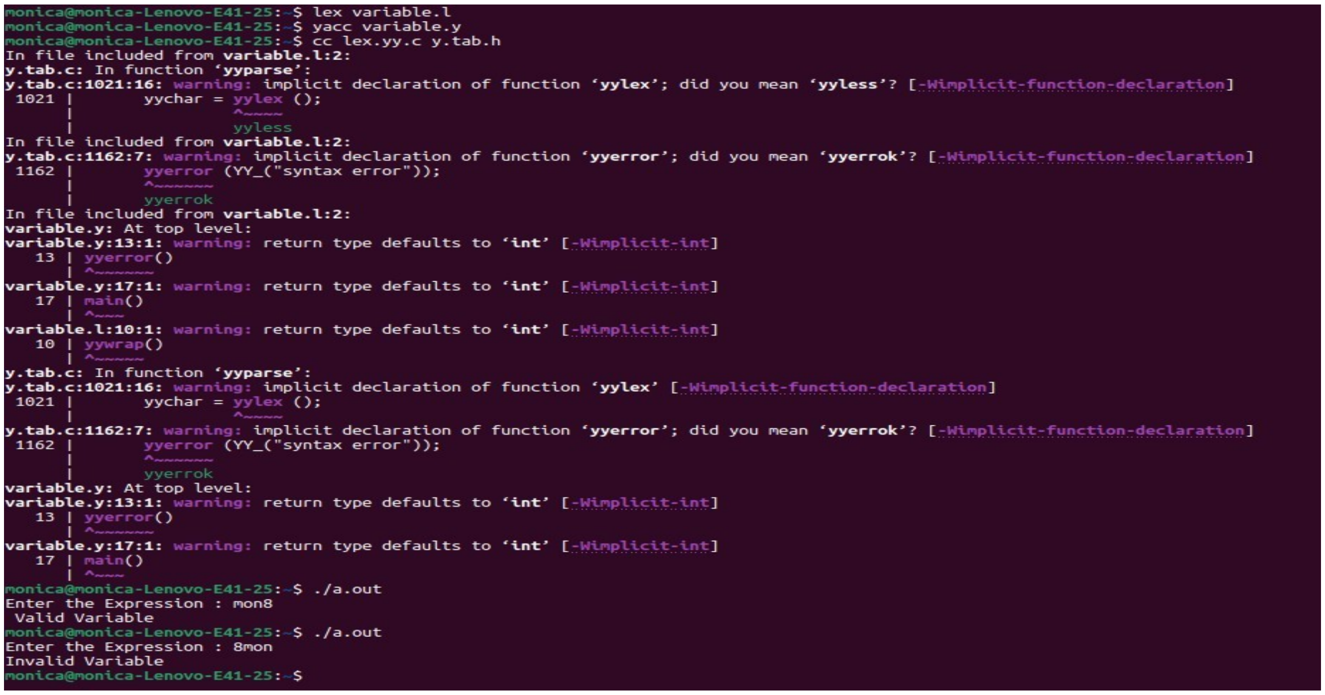
```
%{
 #include "y.tab.h"
%}
%%
[a-zA-Z] { return ALPHA ;}
[0-9]+ { return NUMBER ;
}
"\n" { return ENTER ;}
. { return ER; }
%%
yywrap()
{}
```

YACC PART : variable.y

```
%{
#include<stdio.h>
#include<stdlib.h>
%}
%token ALPHA NUMBER ENTER ER
%%
var : v ENTER  { printf(" Valid Variable\n"); exit(0);}
v:ALPHA exp1
exp1:ALPHA exp1
|NUMBER exp1
|;
%%
yyerror()
{printf("Invalid Variable\n");}
main()
{
printf("Enter the Expression : ");
yyparse();
}
```

OUTPUT:

$:lex variable.l
$:yacc variable.y
$:cc lex.yy.c y.tab.h
$:./a.out



```
monica@monica-Lenovo-E41-25:~$ lex variable.l
monica@monica-Lenovo-E41-25:~$ yacc variable.y
monica@monica-Lenovo-E41-25:~$ cc lex.yy.c y.tab.h
In file included from variable.l:2:
y.tab.c: In function ‘yyparse’:
y.tab.c:1021:16: warning: implicit declaration of function ‘yylex’; did you mean ‘yyless’? [-Wimplicit-function-declaration]
 1021 |        yychar = yylex ();
      |                 ^~~~~
      |                 yyless
In file included from variable.l:2:
y.tab.c:1162:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
 1162 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
In file included from variable.l:2:
variable.y: At top level:
variable.y:13:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   13 | yyerror()
      | ^~~~~~~
variable.y:17:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   17 | main()
      | ^~~~
variable.l:10:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   10 | yywrap()
      | ^~~~~~
y.tab.c: In function ‘yyparse’:
y.tab.c:1021:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
 1021 |        yychar = yylex ();
      |                 ^~~~~
y.tab.c:1162:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
 1162 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
variable.y: At top level:
variable.y:13:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   13 | yyerror()
      | ^~~~~~~
variable.y:17:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   17 | main()
      | ^~~~
monica@monica-Lenovo-E41-25:~$ ./a.out
Enter the Expression : mon8
 Valid Variable
monica@monica-Lenovo-E41-25:~$ ./a.out
Enter the Expression : 8mon
Invalid Variable
monica@monica-Lenovo-E41-25:~$
```

PROGRAM: **FOR LOOP**

LEX PART : for.l

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
alpha [A-Za-z]
digit [0-9]
%%
[\t \n]
for             return FOR;
{digit}+    return NUM;
{alpha}({alpha}|{digit})* return ID;
"<="         return LE;
">="         return GE;
"=="         return EQ;
"!="          return NE;
"||"           return OR;
"&&"          return AND;
.                return yytext[0];
%%
yywrap()
{}
```

YACC PART : for.y

```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%token ID NUM FOR LE GE EQ NE OR AND
%right '='
%left OR AND
%left '>' '<' LE GE EQ NE
%left '+' '-'
%left '*' '/'
%right UMINUS
%left '!'
%%
S         : ST {printf("Input accepted\n"); exit(0);};
ST       : FOR '(' E ';' E2 ';' E ')' DEF
           ;
DEF    : '{' BODY '}'
          | E';'
          | ST
          |
           ;
BODY  : BODY BODY
           | E ';'
           | ST
           |
           ;
E         : ID '=' E
          | E '+' E
          | E '-' E
          | E '*' E
          | E '/' E
          | E '<' E
          | E '>' E
          | E LE E
          | E GE E
          | E EQ E
          | E NE E
          | E OR E
          | E AND E
          | E '+' '+'
          | E '-' '-'
          | ID
          | NUM
          ;
E2     : E'<'E
         | E'>'E
         | E LE E
         | E GE E
         | E EQ E
         | E NE E
         | E OR E
         | E AND E
         ;
%%
main() {
    printf("Enter the expression:\n");
    yyparse();
}
yyerror()
{
   printf("\nEntered arithmetic expression is Invalid\n\n");
}
```

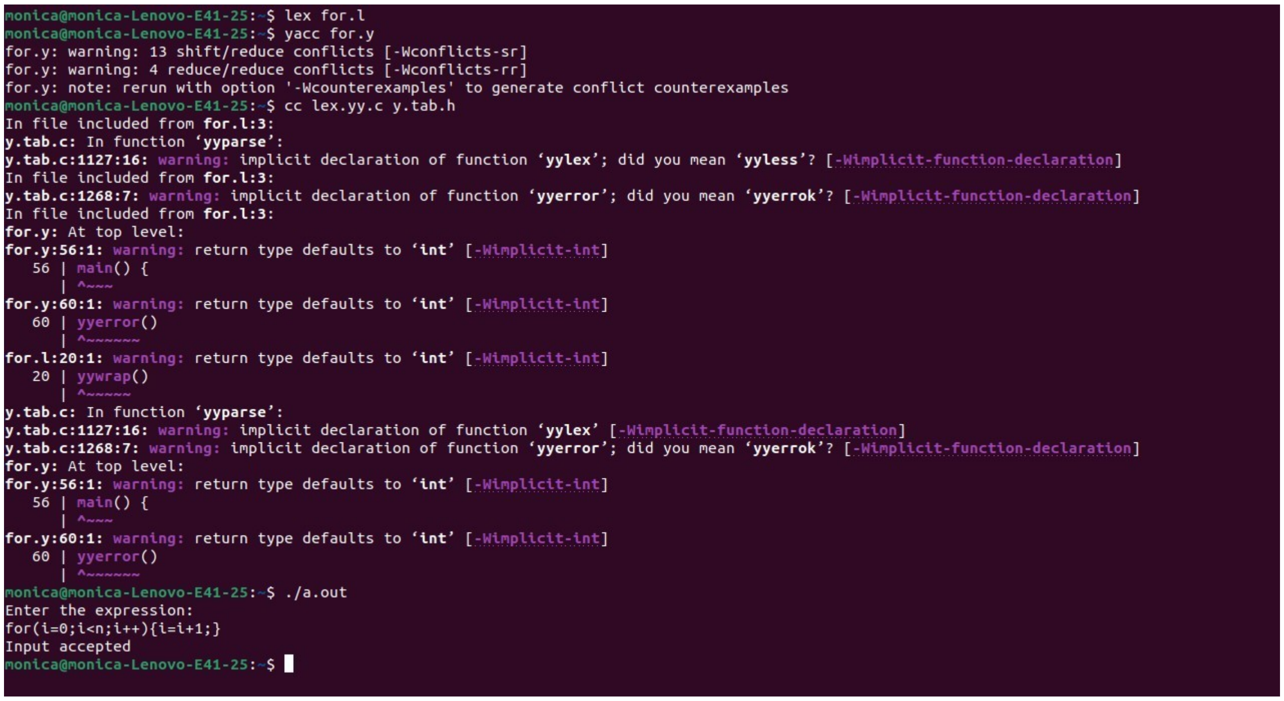
OUTPUT:

$:lex for.l

$:yacc for.y

$:cc lex.yy.c y.tab.h

$:./a.out



```
monica@monica-Lenovo-E41-25:~$ lex for.l
monica@monica-Lenovo-E41-25:~$ yacc for.y
for.y: warning: 13 shift/reduce conflicts [-Wconflicts-sr]
for.y: warning: 4 reduce/reduce conflicts [-Wconflicts-rr]
for.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
monica@monica-Lenovo-E41-25:~$ cc lex.yy.c y.tab.h
In file included from for.l:3:
y.tab.c: In function ‘yyparse’:
y.tab.c:1127:16: warning: implicit declaration of function ‘yylex’; did you mean ‘yyless’? [-Wimplicit-function-declaration]
In file included from for.l:3:
y.tab.c:1268:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
In file included from for.l:3:
for.y: At top level:
for.y:56:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   56 | main() {
      | ^~~~
for.y:60:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   60 | yyerror()
      | ^~~~~~~
for.l:20:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   20 | yywrap()
      | ^~~~~~
y.tab.c: In function ‘yyparse’:
y.tab.c:1127:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
y.tab.c:1268:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
for.y: At top level:
for.y:56:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   56 | main() {
      | ^~~~
for.y:60:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   60 | yyerror()
      | ^~~~~~~
monica@monica-Lenovo-E41-25:~$ ./a.out
Enter the expression:
for(i=0;i<n;i++){i=i+1;}
Input accepted
monica@monica-Lenovo-E41-25:~$
```

PROGRAM: **WHILE LOOP**

LEX PART : while.l

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
alpha [A-Za-z]
digit [0-9]
%%
[\t \n]
while           return WHILE;
{digit}+    return NUM;
{alpha}({alpha}|{digit})* return ID;
"<="        return LE;
">="        return GE;
"=="        return EQ;
"!="        return NE;
"||"        return OR;
"&&"         return AND;
.           return yytext[0];
%%
yywrap()
{}
```

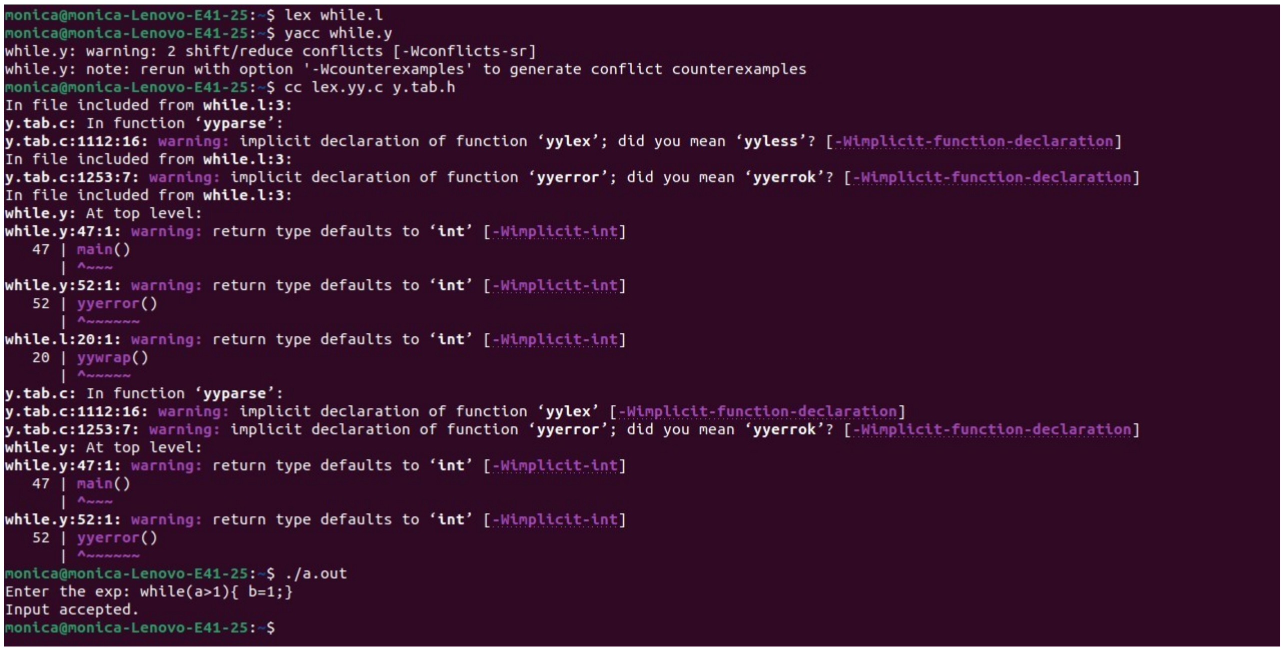
YACC PART :while.y

```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%token ID NUM WHILE LE GE EQ NE OR AND
%right '='
%left AND OR
%left '<' '>' LE GE EQ NE
%left '+''-'
%left '*''/'
%right UMINUS
%left '!'
%%
S       : ST1 {printf("Input accepted.\n");exit(0);};
ST1    :    WHILE'(' E2 ')' '{' ST
'}' ST  :     ST ST
          | E';'
          ;
E       : ID'='E
          | E'+'E
          | E'-'E
          | E'*'E
          | E'/'E
          | E'<'E
          | E'>'E
          | E LE E
          | E GE E
          | E EQ E
          | E NE E
          | E OR E
          | E AND E
          | ID
          | NUM
          ;
E2     : E'<'E
          | E'>'E
          | E LE E
          | E GE E
          | E EQ E
          | E NE E
          | E OR E
          | E AND E
          | ID
```

```
        | NUM
        ;
%%
main()
{
  printf("Enter the exp: ");
  yyparse();
}
yyerror()
{
  printf("\nEntered arithmetic expression is Invalid\n\n");
}
```

OUTPUT:

```
$:lex while.l
$:yacc while.y
$:cc lex.yy.c y.tab.h
$:./a.out
```



```
monica@monica-Lenovo-E41-25:~$ lex while.l
monica@monica-Lenovo-E41-25:~$ yacc while.y
while.y: warning: 2 shift/reduce conflicts [-Wconflicts-sr]
while.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
monica@monica-Lenovo-E41-25:~$ cc lex.yy.c y.tab.h
In file included from while.l:3:
y.tab.c: In function ‘yyparse’:
y.tab.c:1112:16: warning: implicit declaration of function ‘yylex’; did you mean ‘yyless’? [-Wimplicit-function-declaration]
In file included from while.l:3:
y.tab.c:1253:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
In file included from while.l:3:
while.y: At top level:
while.y:47:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   47 | main()
      | ^~~~
while.y:52:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   52 | yyerror()
      | ^~~~~~~
while.l:20:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   20 | yywrap()
      | ^~~~~~
y.tab.c: In function ‘yyparse’:
y.tab.c:1112:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
y.tab.c:1253:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
while.y: At top level:
while.y:47:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   47 | main()
      | ^~~~
while.y:52:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   52 | yyerror()
      | ^~~~~~~
monica@monica-Lenovo-E41-25:~$ ./a.out
Enter the exp: while(a>1){ b=1;}
Input accepted.
monica@monica-Lenovo-E41-25:~$
```

PROGRAM: **IF THEN ELSE**

LEX PART : if.l

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
alpha [A-Za-z]
digit [0-9]
%%
[ \t\n]
```

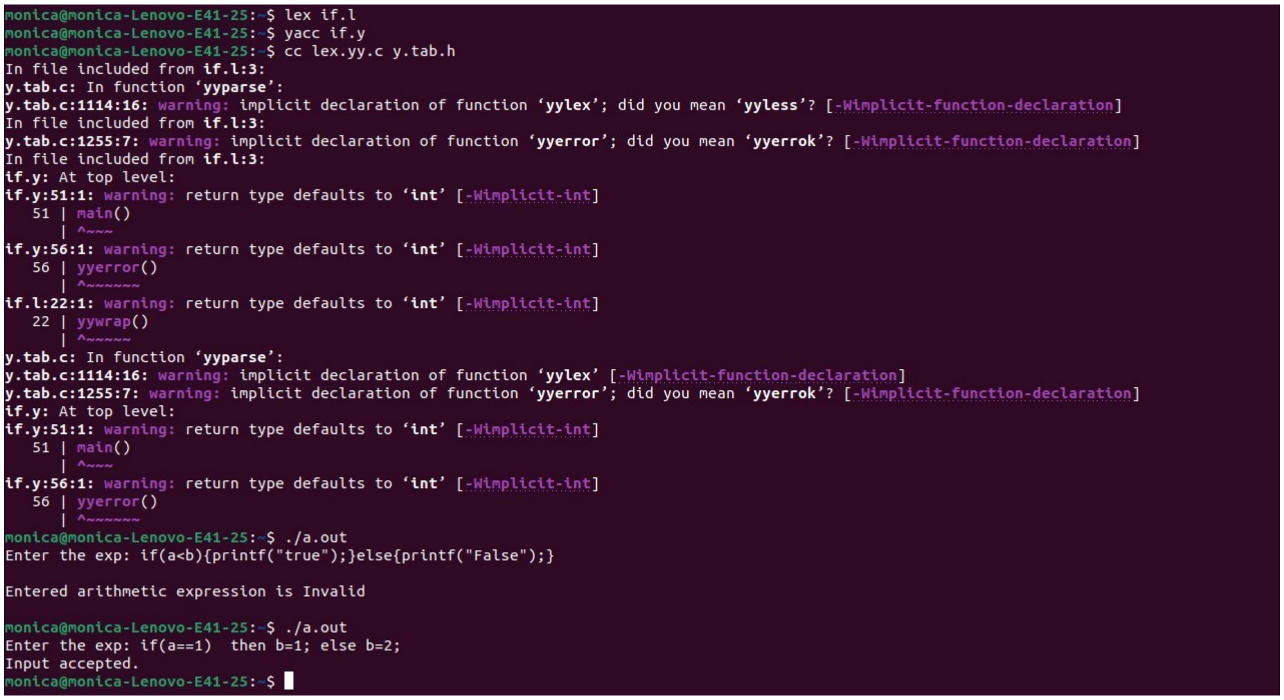
```
if    return IF;
then    return THEN;
else    return ELSE;
{digit}+    return NUM;
{alpha}({alpha}|{digit})*    return ID;
"<="    return LE;
">="   return   GE;
"=="   return   EQ;
"!="   return   NE;
"||"  return OR;
"&&"    return AND;
.    return yytext[0];
%%
yywrap()
{}
```

YACC PART : if.y

```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%token ID NUM IF THEN LE GE EQ NE OR AND ELSE
%right '='
%left AND OR
%left '<' '>' LE GE EQ NE
%left '+''-'
%left '*''/'
%right UMINUS
%left '!'
%%
S      : ST {printf("Input accepted.\n");exit(0);};
ST     : IF '(' E2 ')' THEN ST1';' ELSE ST1';'
        | IF '(' E2 ')' THEN ST1';'
        ;
ST1  : ST
        | E
        ;
E    : ID'='E
      | E'+'E
      | E'-'E
      | E'*'E
      | E'/'E
      | E'<'E
      | E'>'E
      | E LE E
      | E GE E
      | E EQ E
      | E NE E
      | E OR E
      | E AND E
      | ID
      | NUM
      ;
E2  : E'<'E
      | E'>'E
      | E LE E
      | E GE E
      | E EQ E
      | E NE E
      | E OR E
      | E AND E
      | ID
      | NUM
      ;
%%
main()
{
  printf("Enter the exp: ");
  yyparse();
}
yyerror()
{
   printf("\nEntered arithmetic expression is Invalid\n\n");
}
```

OUTPUT:

```
$:lex if.l
$:yacc if.y
$:cc lex.yy.c y.tab.h
$:./a.out
```



```
monica@monica-Lenovo-E41-25:~$ lex if.l
monica@monica-Lenovo-E41-25:~$ yacc if.y
monica@monica-Lenovo-E41-25:~$ cc lex.yy.c y.tab.h
In file included from if.l:3:
y.tab.c: In function ‘yyparse’:
y.tab.c:1114:16: warning: implicit declaration of function ‘yylex’; did you mean ‘yyless’? [-Wimplicit-function-declaration]
In file included from if.l:3:
y.tab.c:1255:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
In file included from if.l:3:
if.y: At top level:
if.y:51:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   51 | main()
      | ^~~~
if.y:56:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   56 | yyerror()
      | ^~~~~~~
if.l:22:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   22 | yywrap()
      | ^~~~~~
y.tab.c: In function ‘yyparse’:
y.tab.c:1114:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
y.tab.c:1255:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
if.y: At top level:
if.y:51:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   51 | main()
      | ^~~~
if.y:56:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   56 | yyerror()
      | ^~~~~~~
monica@monica-Lenovo-E41-25:~$ ./a.out
Enter the exp: if(a<b){printf("true");}else{printf("False");}

Entered arithmetic expression is Invalid

monica@monica-Lenovo-E41-25:~$ ./a.out
Enter the exp: if(a==1)  then b=1; else b=2;
Input accepted.
monica@monica-Lenovo-E41-25:~$ █
```

PROGRAM: **IF THEN ELSEIF THEN ELSE**

LEX PART : elseif.l

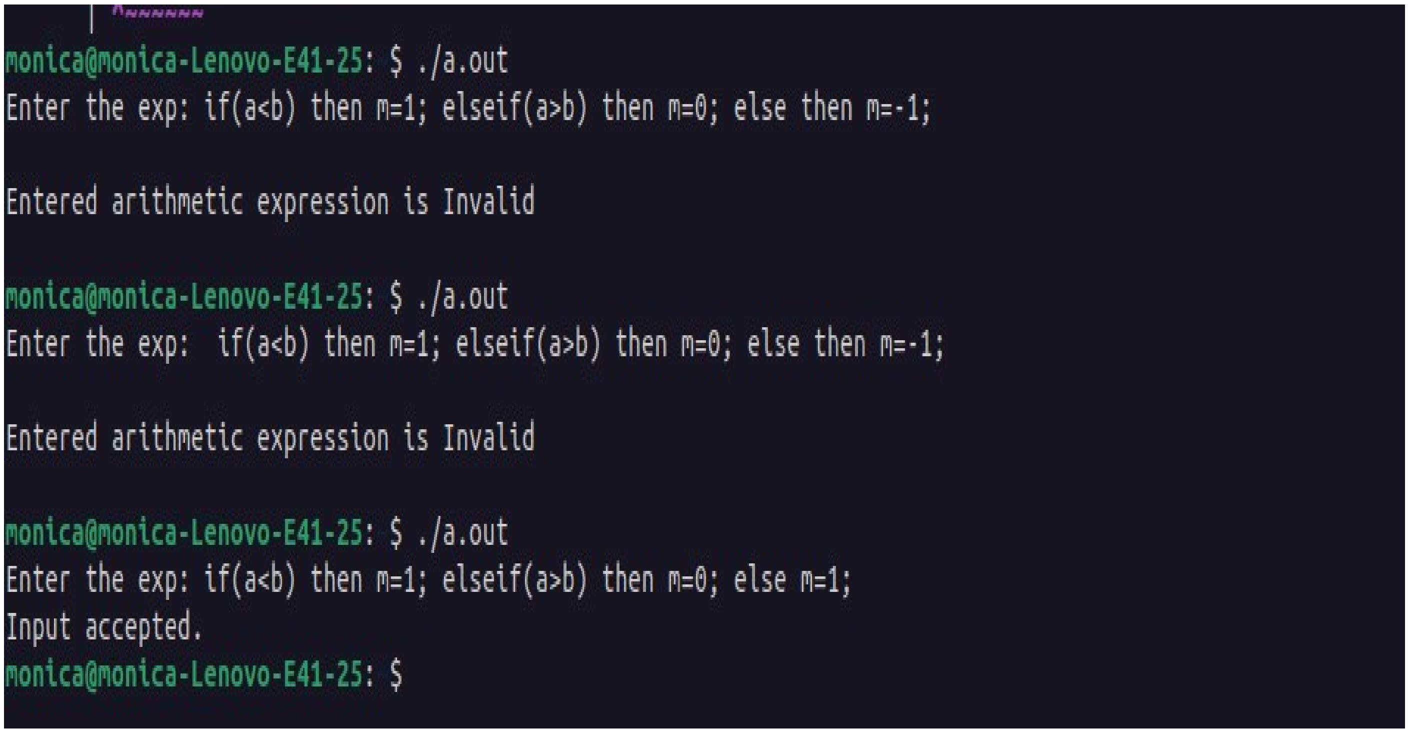
```
%{
#include<stdio.h>
#include "y.tab.h"
%}
alpha [A-Za-z]
digit [0-9]
%%
[ \t\n]
if    return IF;
then    return THEN;
else    return ELSE;
elseif   return ELSEIF;
{digit}+    return NUM;
{alpha}({alpha}|{digit})*   return   ID;
"<="  return LE;
">="   return   GE;
"=="   return   EQ;
"!="   return   NE;
"||"  return OR;
"&&"    return AND;
.    return yytext[0];
%%
yywrap()
{}
```

YACC PART : elseif.y

```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%token ID NUM IF THEN LE GE EQ NE OR AND ELSE ELSEIF
%right '='
%left AND OR
%left '<' '>' LE GE EQ NE
%left '+''-'
%left '*''/'
%right UMINUS
%left '!'
%%
S      : ST {printf("Input accepted.\n");exit(0);};
ST    : IF '(' E2 ')' THEN ST1';' ELSEIF '(' E2 ')' THEN ST1';' ELSE ST1';'
        | IF '(' E2 ')' THEN ST1';'
        ;
ST1  : ST
        | E
        ;
E    : ID'='E
      | E'+'E
      | E'-'E
      | E'*'E
      | E'/'E
      | E'<'E
      | E'>'E
      | E LE E
      | E GE E
      | E EQ E
      | E NE E
      | E OR E
      | E AND E
      | ID
      | NUM
      ;
E2  : E'<'E
      | E'>'E
      | E LE E
      | E GE E
      | E EQ E
      | E NE E
      | E OR E
      | E AND E
      | ID
      | NUM
      ;
%%
main()
{
  printf("Enter the exp: ");
  yyparse();
}
yyerror()
{
   printf("\nEntered arithmetic expression is Invalid\n\n");
}
```

OUTPUT:

```
$:lex elseif.l
$:yacc elseif.y
$:cc lex.yy.c y.tab.h
$:./a.out
```



```
monica@monica-Lenovo-E41-25: $ ./a.out
Enter the exp: if(a<b) then m=1; elseif(a>b) then m=0; else then m=-1;

Entered arithmetic expression is Invalid

monica@monica-Lenovo-E41-25: $ ./a.out
Enter the exp:  if(a<b) then m=1; elseif(a>b) then m=0; else then m=-1;

Entered arithmetic expression is Invalid

monica@monica-Lenovo-E41-25: $ ./a.out
Enter the exp: if(a<b) then m=1; elseif(a>b) then m=0; else m=1;
Input accepted.
monica@monica-Lenovo-E41-25: $
```

PROGRAM: **SWITCH**

LEX PART : switch.l

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
alpha [A-Za-z]
digit [0-9]
%%
[ \n\t]
if                  return IF;
then                return THEN;
while               return WHILE;
switch              return SWITCH;
case                return CASE;
default             return DEFAULT;
break               return BREAK;
{digit}+            return NUM;
{alpha}({alpha}|{digit})* return ID;
"<="                return LE;
">="                return GE;
"=="                return EQ;
"!="                return NE;
"&&"                return AND;
"||"                return OR;
.                   return yytext[0];
%%
yywrap()
{}
```

YACC PART : switch.y

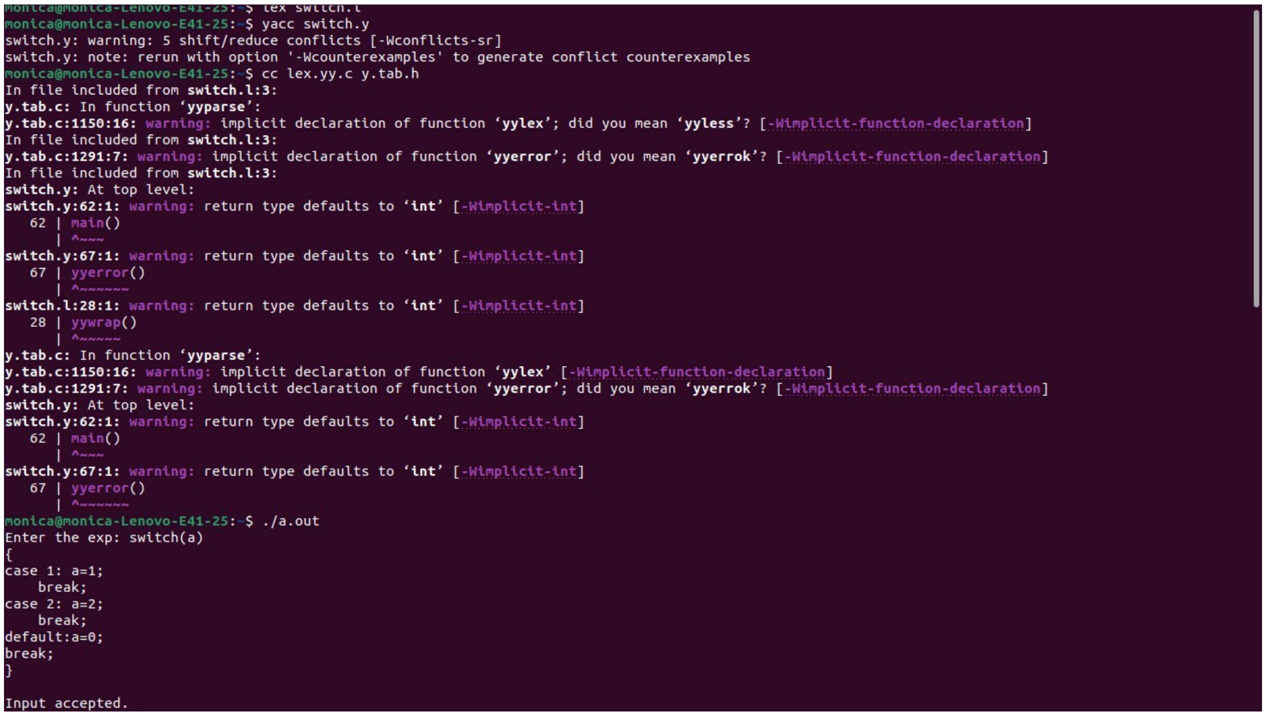
```
%{
#include<stdio.h>
#include<stdlib.h>
%}
%token ID NUM SWITCH CASE DEFAULT BREAK LE GE EQ NE AND OR IF THEN WHILE
%right '='
%left AND OR
%left '<' '>' LE GE EQ NE
%left '+''-'
%left '*''/'
%right UMINUS
%left '!'
%%
S    :    ST{printf("\nInput accepted.\n");exit(0);};
     ;
ST   :    SWITCH'('ID')''{'B'}'
     ;
B    :    C
     |    C D
```

```
      ;
C  :    C C
    |    CASE NUM':'ST1 BREAK';'
    ;
D  :    DEFAULT':'ST1 BREAK';'
    |    DEFAULT':'ST1
    ;
ST1    :    WHILE'('E2')' E';'
    |    IF'('E2')'THEN E';'
    |    ST1 ST1
    |    E';'
    ;
E2    :    E'<'E
    |    E'>'E
    |    E LE E
    |    E GE E
    |    E EQ E
    |    E NE E
    |    E AND E
    |    E OR E
    ;
E    :    ID'='E
    |    E'+'E
    |    E'-'E
    |    E'*'E
    |    E'/'E
    |    E'<'E
    |    E'>'E
    |    E LE E
    |    E GE E
    |    E EQ E
    |    E NE E
    |    E AND E
    |    E OR E
    |    ID
    |    NUM
    ;
%%
main()
{
  printf("Enter the exp: ");
  yyparse();
}
yyerror()
```

```
{
  printf("\nEntered arithmetic expression is Invalid\n\n");
}
```

OUTPUT:

```
$:lex switch.l
$:yacc switch.y
$:cc lex.yy.c y.tab.h
$:./a.out
```



```
monica@monica-Lenovo-E41-25:~$ lex switch.l
monica@monica-Lenovo-E41-25:~$ yacc switch.y
switch.y: warning: 5 shift/reduce conflicts [-Wconflicts-sr]
switch.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
monica@monica-Lenovo-E41-25:~$ cc lex.yy.c y.tab.h
In file included from switch.l:3:
y.tab.c: In function ‘yyparse’:
y.tab.c:1150:16: warning: implicit declaration of function ‘yylex’; did you mean ‘yyless’? [-Wimplicit-function-declaration]
In file included from switch.l:3:
y.tab.c:1291:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
In file included from switch.l:3:
switch.y: At top level:
switch.y:62:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   62 | main()
      | ^~~~
switch.y:67:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   67 | yyerror()
      | ^~~~~~~
switch.l:28:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   28 | yywrap()
      | ^~~~~~
y.tab.c: In function ‘yyparse’:
y.tab.c:1150:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
y.tab.c:1291:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
switch.y: At top level:
switch.y:62:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   62 | main()
      | ^~~~
switch.y:67:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   67 | yyerror()
      | ^~~~~~~
monica@monica-Lenovo-E41-25:~$ ./a.out
Enter the exp: switch(a)
{
case 1: a=1;
    break;
case 2: a=2;
    break;
default:a=0;
break;
}

Input accepted.
```

PROGRAM:

LEX PART : calc.l

```
%{
        #include<stdlib.h>
        #include "y.tab.h"
        extern int yylval;
%}
%%
[0-9]+ {yylval=atoi(yytext); return NUMBER;}
[\n] return 0;
[\t];
. return yytext[0];
%%
yywrap()
{ return 0;}
```

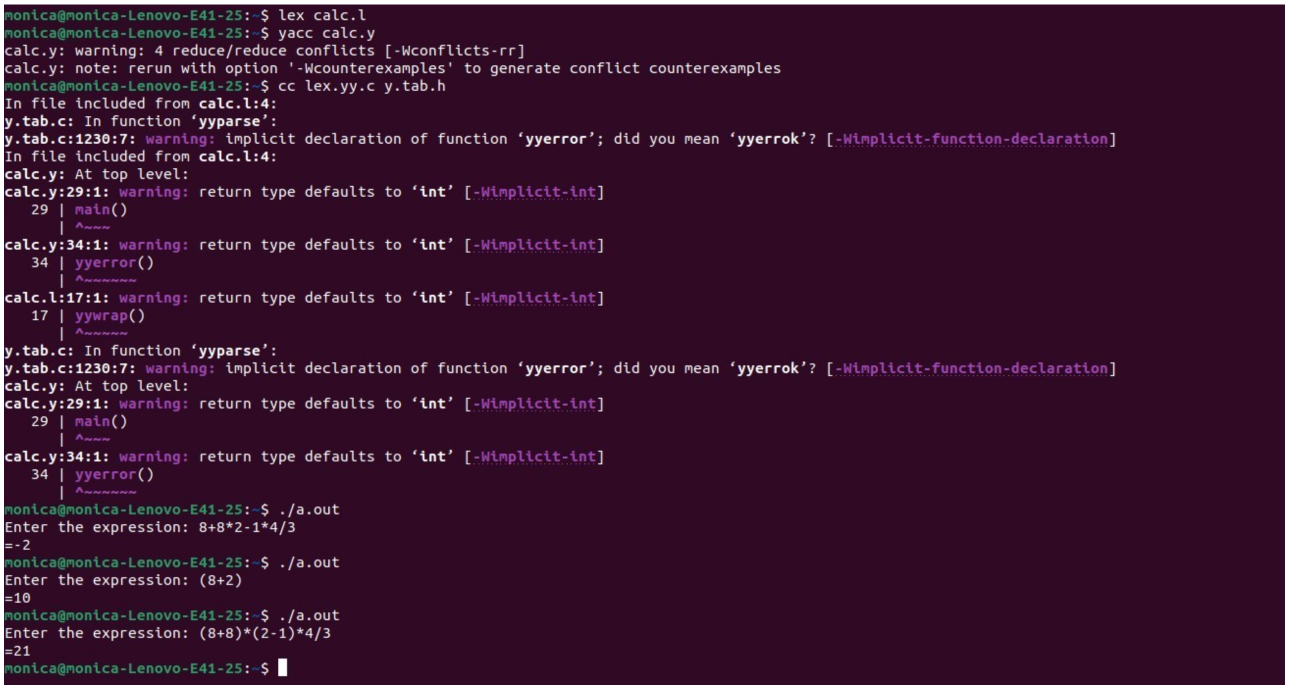
YACC PART : calc.y

```
%{
        #include<stdio.h>
        int yylex(void);
%}
%token NAME NUMBER
%left GE LE NE EQ '<' '>' '%'
%left '-' '+'
%left '*' '/'
%nonassoc UMINUS
%%
statement : NAME '=' exp
        |exp {printf("=%d\n",$1);}
        ;
exp : NUMBER {$$ == $1;}
        |exp '+' exp {$$ = $1 + $3 ;}
        |exp '-' exp {$$ = $1 - $3 ;}
        |exp '*' exp {$$ = $1 * $3 ;}
        |exp '/' exp {$$ = $1 / $3 ;}
        |exp '-' exp %prec UMINUS {$$ = -$2 ;}
        |'(' exp ')' {$$ = $2 ;}
;
%%
main()
{
printf("Enter the expression: ");
yyparse();
}
yyerror()
```

```
{ printf("\nError Occured\n");
}
```

OUTPUT:

```
$:lex calc.l
$:yacc calc.y
$:cc lex.yy.c y.tab.h
$:./a.out
```



```
monica@monica-Lenovo-E41-25:~$ lex calc.l
monica@monica-Lenovo-E41-25:~$ yacc calc.y
calc.y: warning: 4 reduce/reduce conflicts [-Wconflicts-rr]
calc.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
monica@monica-Lenovo-E41-25:~$ cc lex.yy.c y.tab.h
In file included from calc.l:4:
y.tab.c: In function ‘yyparse’:
y.tab.c:1230:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
In file included from calc.l:4:
calc.y: At top level:
calc.y:29:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   29 | main()
      | ^~~~
calc.y:34:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   34 | yyerror()
      | ^~~~~~~
calc.l:17:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   17 | yywrap()
      | ^~~~~~
y.tab.c: In function ‘yyparse’:
y.tab.c:1230:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
calc.y: At top level:
calc.y:29:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   29 | main()
      | ^~~~
calc.y:34:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   34 | yyerror()
      | ^~~~~~~
monica@monica-Lenovo-E41-25:~$ ./a.out
Enter the expression: 8+8*2-1*4/3
=-2
monica@monica-Lenovo-E41-25:~$ ./a.out
Enter the expression: (8+2)
=10
monica@monica-Lenovo-E41-25:~$ ./a.out
Enter the expression: (8+8)*(2-1)*4/3
=21
monica@monica-Lenovo-E41-25:~$ 
```

PROGRAM:

LEX PART: lex three.l

```
%{
#include"y.tab.h"
extern char yyval;
%}
%%
[0-9]+ {yylval.symbol=(char)(yytext[0]);return NUMBER;}
[a-z] {yylval.symbol= (char)(yytext[0]);return LETTER;}
. {return yytext[0];}
\n {return 0;}
%%
```

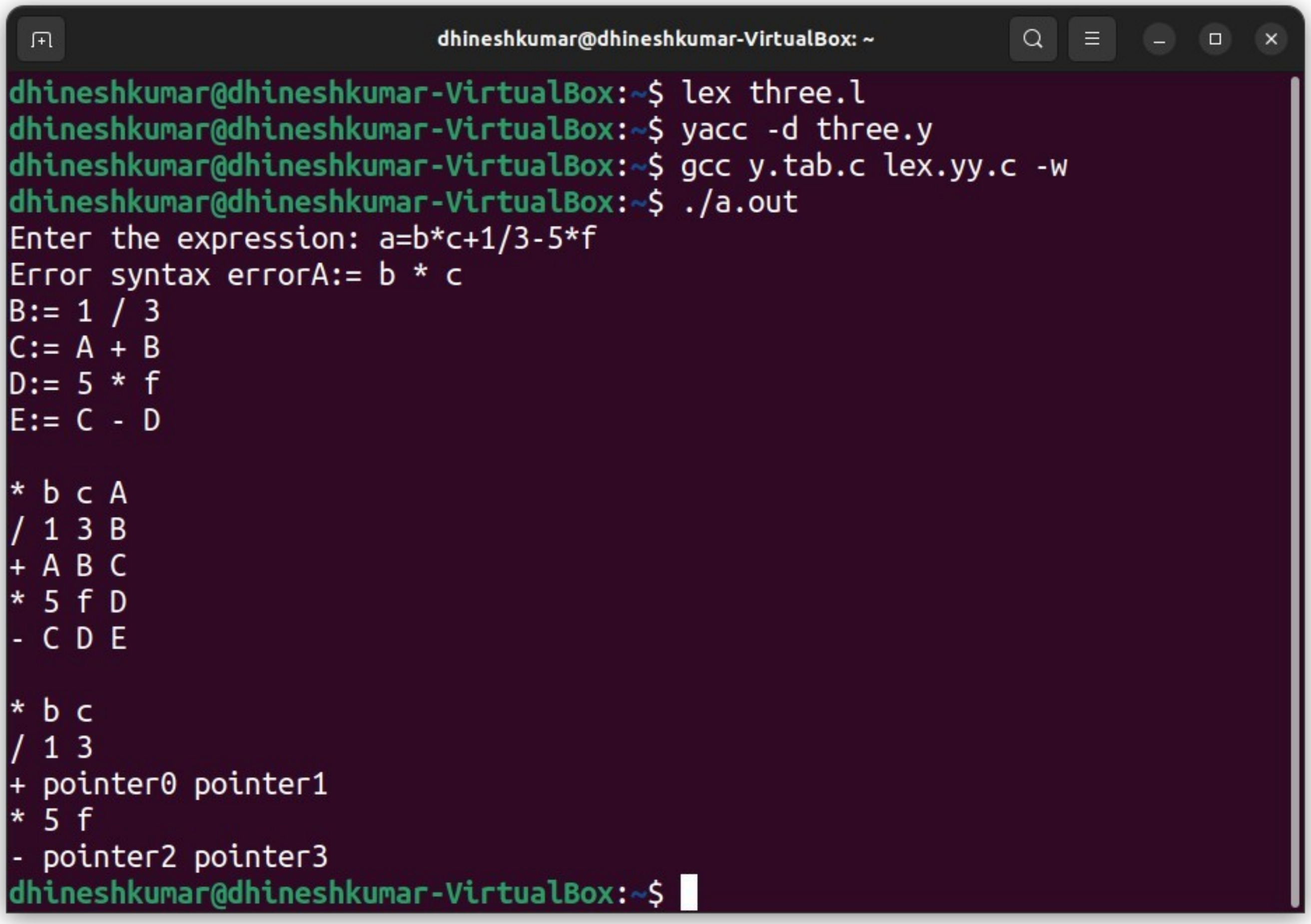
YACC PART: yacc three.y

```
%{
#include "y.tab.h"
#include <stdio.h>
char addtotable(char, char, char);
int index1 = 0;
char temp = 'A' - 1;
struct expr {
   char operand1;
   char operand2;
   char operator;
   char result;
};
%}
%union {
   char symbol;
}
%left '+' '-'
%left '/' '*'
%token <symbol> LETTER NUMBER
%type <symbol> exp
%%
statement: LETTER '=' exp ';' {  addtotable((char)$1, (char)$3, '=');};
exp: exp '+' exp {    $$ = addtotable((char)$1, (char)$3, '+');}
| exp '-' exp {    $$ = addtotable((char)$1, (char)$3, '-');}
| exp '/' exp {    $$ = addtotable((char)$1, (char)$3, '/');}
| exp '*' exp {    $$ = addtotable((char)$1, (char)$3, '*');}
| '(' exp ')' {    $$ = (char)$2;}
| NUMBER {    $$ = (char)$1;}
| LETTER {    (char)$1;}
%%
```

```
struct expr arr[20];
void yyerror(char *s) {    printf("Error %s", s);}
char addtotable(char a, char b, char o) {
    temp++;
    arr[index1].operand1 = a;
    arr[index1].operand2 = b;
    arr[index1].operator = o;
    arr[index1].result = temp;
    index1++;
    return temp;
}
void threeAdd() {
    int i = 0;
    char temp = 'A';
    while (i < index1) {
        printf("%c := ", arr[i].result);
        printf("%c ", arr[i].operand1);
        printf("%c ", arr[i].operator);
        printf("%c ", arr[i].operand2);
        i++;
        temp++;
        printf("\n");
    }
}
void fouradd() {
    int i = 0;
    char temp = 'A';
    while (i < index1) {
        printf("%c ", arr[i].operator);
        printf("%c ", arr[i].operand1);
        printf("%c ", arr[i].operand2);
        printf("%c ", arr[i].result);
        i++;
        temp++;
        printf("\n");
    }
}
int find(char l) {
    int i;
    for (i = 0; i < index1; i++)
        if (arr[i].result == l)
            break;
    return i;
}
```

```
void triple() {
    int i = 0;
    char temp = 'A';
    while (i < index1) {
        printf("%c ", arr[i].operator);
        if (!isupper(arr[i].operand1))
            printf("%c ", arr[i].operand1);
        else {
            printf("pointer");
            printf("%d ", find(arr[i].operand1));
        }
        if (!isupper(arr[i].operand2))
            printf("%c ", arr[i].operand2);
        else {
            printf("pointer");
            printf("%d ", find(arr[i].operand2));
        } i+
        +;
        temp++;
        printf("\n");
    }
}
int yywrap() {
    return 1;
}
int main() {
    printf("Enter the expression: ");
    yyparse();
    threeAdd();
    printf("\n");
    fouradd();
    printf("\n");
    triple();
    return 0;
}
```

OUTPUT:


```
dhineshkumar@dhineshkumar-VirtualBox:~$ lex three.l
dhineshkumar@dhineshkumar-VirtualBox:~$ yacc -d three.y
dhineshkumar@dhineshkumar-VirtualBox:~$ gcc y.tab.c lex.yy.c -w
dhineshkumar@dhineshkumar-VirtualBox:~$ ./a.out
Enter the expression: a=b*c+1/3-5*f
Error syntax errorA:= b * c
B:= 1 / 3
C:= A + B
D:= 5 * f
E:= C - D

* b c A
/ 1 3 B
+ A B C
* 5 f D
- C D E

* b c
/ 1 3
+ pointer0 pointer1
* 5 f
- pointer2 pointer3
dhineshkumar@dhineshkumar-VirtualBox:~$
```

PROGRAM:

LEX PART: lex type.l

```
%{
#include"y.tab.h"
#include<stdio.h>
#include<string.h>
int LineNo=1;
%}
identifier [a-zA-Z][_a-zA-Z0-9]*
number [0-9]+|([0-9]*\.[0-9]+)
%%
main\(\) return MAIN;
if return IF;
else return ELSE;
while return WHILE;
int |
char |
float return TYPE;
{identifier} {strcpy(yylval.var,yytext);
return VAR;}
{number} {strcpy(yylval.var,yytext);
return NUM;}
\< |
\> |
\>= |
\<= |
== {strcpy(yylval.var,yytext);return RELOP;}
[ \t] ;
\n LineNo++;
. return yytext[0];
%%
```

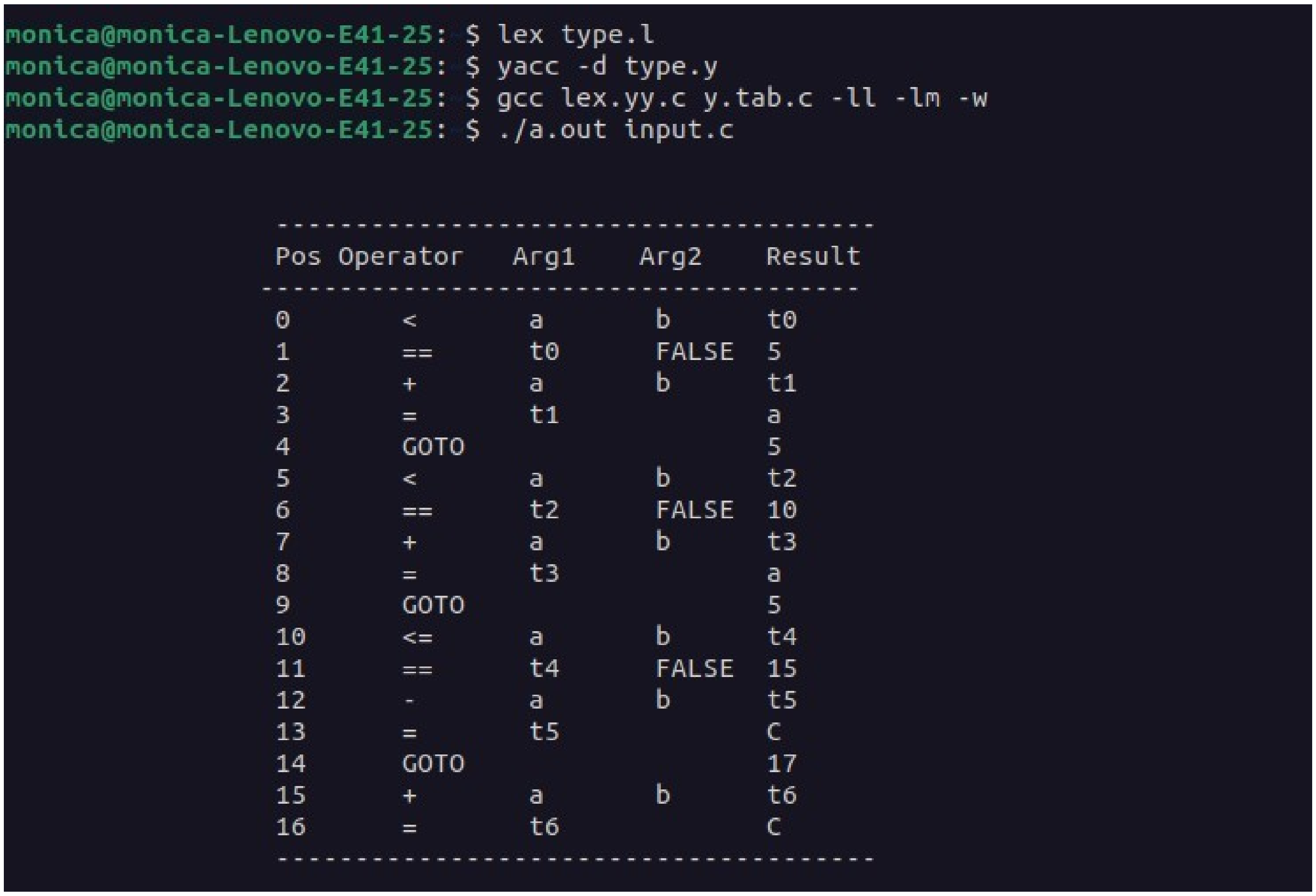
YACC PART :yacc type.c

```
%{
#include<string.h>
#include<stdio.h>
struct quad
{
char op[5];
char arg1[10];
char arg2[10];
char result[10];
}QUAD[30];
struct stack
{
```

```
int items[100];
int top;
}stk;
int Index=0,tIndex=0,StNo,Ind,tInd;
extern int LineNo;
%}
%union
{    char var[10];}
%token <var> NUM VAR RELOP
%token MAIN IF ELSE WHILE TYPE
%type <var> EXPR ASSIGNMENT CONDITION IFST ELSEST WHILELOOP
%left '-' '+'
%left '*' '/'
%%
PROGRAM : MAIN BLOCK
;
BLOCK: '{' CODE '}'
;
CODE: BLOCK
| STATEMENT CODE
| STATEMENT
;
STATEMENT: DESCT ';'
| ASSIGNMENT ';'| CONDST
| WHILEST
;
DESCT: TYPE VARLIST
;
VARLIST: VAR ',' VARLIST
| VAR
;
ASSIGNMENT: VAR '=' EXPR{
strcpy(QUAD[Index].op,"=");
strcpy(QUAD[Index].arg1,$3);
strcpy(QUAD[Index].arg2,"");
strcpy(QUAD[Index].result,$1);
strcpy($$,QUAD[Index++].result);
}
;
EXPR: EXPR '+' EXPR {AddQuadruple("+",$1,$3,$$);}
| EXPR '-' EXPR {AddQuadruple("-",$1,$3,$$);}
| EXPR '*' EXPR {AddQuadruple("*",$1,$3,$$);}
| EXPR '/' EXPR {AddQuadruple("/",$1,$3,$$);}
| '-' EXPR {AddQuadruple("UMIN",$2,"",$$);}
| '(' EXPR ')' {strcpy($$,$2);}
| VAR
| NUM
;
CONDST: IFST{
Ind=pop();
sprintf(QUAD[Ind].result,"%d",Index);
Ind=pop();
sprintf(QUAD[Ind].result,"%d",Index);
}
| IFST ELSEST
;
IFST: IF '(' CONDITION ')' {
strcpy(QUAD[Index].op,"==");
strcpy(QUAD[Index].arg1,$3);
strcpy(QUAD[Index].arg2,"FALSE");
strcpy(QUAD[Index].result,"-1");
push(Index);
Index++;
}
BLOCK { strcpy(QUAD[Index].op,"GOTO"); strcpy(QUAD[Index].arg1,"");
strcpy(QUAD[Index].arg2,"");
strcpy(QUAD[Index].result,"-1");
push(Index);
Index++;};
ELSEST: ELSE{
tInd=pop();
Ind=pop();
push(tInd);
sprintf(QUAD[Ind].result,"%d",Index);
}
BLOCK{
Ind=pop();
sprintf(QUAD[Ind].result,"%d",Index);
};
CONDITION: VAR RELOP VAR {AddQuadruple($2,$1,$3,$
$); StNo=Index-1;
}
| VAR
| NUM
;
WHILEST: WHILELOOP{
Ind=pop();
sprintf(QUAD[Ind].result,"%d",StNo);
```

```
Ind=pop();
sprintf(QUAD[Ind].result,"%d",Index);
}
;
WHILELOOP: WHILE'('CONDITION ')' {
strcpy(QUAD[Index].op,"==");
strcpy(QUAD[Index].arg1,$3);
strcpy(QUAD[Index].arg2,"FALSE");
strcpy(QUAD[Index].result,"-1");
push(Index);
Index++;
}
BLOCK {
strcpy(QUAD[Index].op,"GOTO");
strcpy(QUAD[Index].arg1,"");
strcpy(QUAD[Index].arg2,"");
strcpy(QUAD[Index].result,"-1");
push(Index);
Index++;
}
;
%%
extern FILE *yyin;
int main(int argc,char *argv[])
{
FILE *fp;int i;
if(argc>1)
{
fp=fopen(argv[1],"r"); if(!
fp)
{
printf("\n File not found");
exit(0);
}
yyin=fp;
}
yyparse();
printf("\n\n\t\t----------------------------""\n\t\t Pos Operator \tArg1 \tArg2 \tResult" "\n\t\
t-------------------");
for(i=0;i<Index;i++)
{
printf("\n\t\t %d\t %s\t %s\t %s\t%s",i,QUAD[i].op,QUAD[i].arg1,QUAD[i].arg2,QUAD[i].result);
}
printf("\n\t\t-----------------------");
printf("\n\n"); return 0; }
void push(int data)
{ stk.top++;
if(stk.top==100)
{
printf("\n Stack overflow\n");
exit(0);
}
stk.items[stk.top]=data;
}
int pop()
{
int data;
if(stk.top==-1)
{
printf("\n Stack underflow\n");
exit(0);
}
data=stk.items[stk.top--];
return data;
}
void AddQuadruple(char op[5],char arg1[10],char arg2[10],char result[10])
{
strcpy(QUAD[Index].op,op);
strcpy(QUAD[Index].arg1,arg1);
strcpy(QUAD[Index].arg2,arg2);
sprintf(QUAD[Index].result,"t%d",tIndex++);strcpy(result,QUAD[Index++].result);
}
yyerror()
{
printf("\n Error on line no:%d",LineNo);
}
```

OUTPUT:



```
monica@monica-Lenovo-E41-25: $ lex type.l
monica@monica-Lenovo-E41-25: $ yacc -d type.y
monica@monica-Lenovo-E41-25: $ gcc lex.yy.c y.tab.c -ll -lm -w
monica@monica-Lenovo-E41-25: $ ./a.out input.c


                  --------------------------------------
                  Pos Operator   Arg1    Arg2    Result
                 --------------------------------------
                  0       <       a       b       t0
                  1       ==      t0      FALSE   5
                  2       +       a       b       t1
                  3       =       t1              a
                  4       GOTO                    5
                  5       <       a       b       t2
                  6       ==      t2      FALSE   10
                  7       +       a       b       t3
                  8       =       t3              a
                  9       GOTO                    5
                  10      <=      a       b       t4
                  11      ==      t4      FALSE   15
                  12      -       a       b       t5
                  13      =       t5              C
                  14      GOTO                    17
                  15      +       a       b       t6
                  16      =       t6              C
                  --------------------------------------
```

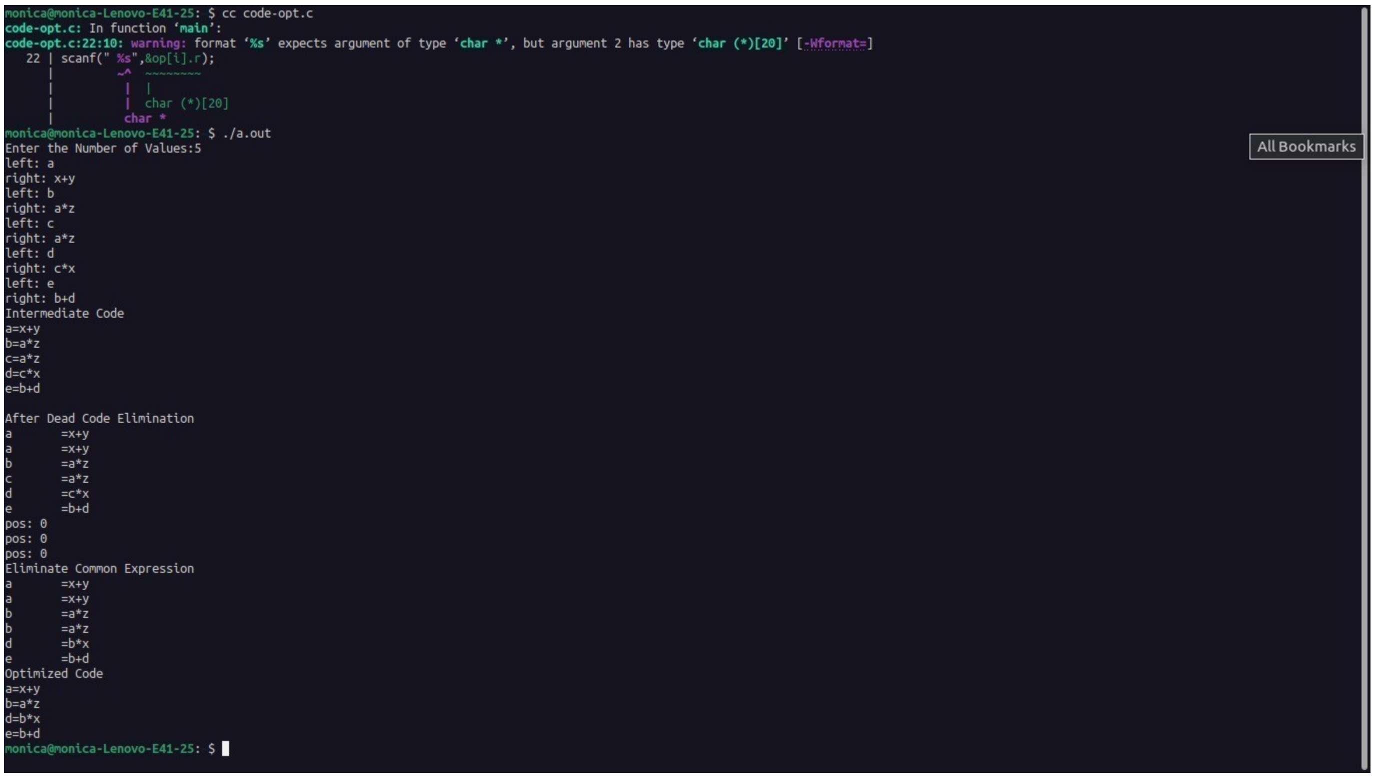
PROGRAM:

```
#include<stdio.h>
#include<string.h>
struct op
{
char l;
char r[20];
}
op[10],pr[10];
void main()
{
int a,i,k,j,n,z=0,m,q;
char *p,*l;char temp,t;
char *tem;
printf(“Enter the number of values:”);
scanf(“%d”,&n);
for(i=0;i<n;i++)
{
printf(“left:  “);
scanf(“ %c”,&op[i].l);
printf(“right: “);
scanf(“ %s”,&op[i].r);
}
printf(“Intermediate code\n”) ;
for(i=0;i<n;i++)
{
printf(“%c=”,op[i].l);
printf(“%s\n”,op[i].r);
}
for(i=0;i<n-1;i++)
{
temp=op[i].l;
for(j=0;j<n;j++)
{
p=strchr(op[j].r,temp);
if(p)
{
pr[z].l=op[i].l;
strcpy(pr[z].r,op[i].r); z+
+;
}
}
}
```

```
pr[z].l=op[n-1].l;
strcpy(pr[z].r,op[n-1].r);
z++;
printf("\nAfter dead code elimination\n");
for(k=0;k<z;k++)
{
printf("%c\t=",pr[k].l);
printf("%s\n",pr[k].r);
}
for(m=0;m<z;m++)
{
tem=pr[m].r;
for(j=m+1;j<z;j++)
{
p=strstr(tem,pr[j].r);
if(p)
{
t=pr[j].l;
pr[j].l=pr[m].l;
for(i=0;i<z;i++)
{
l=strchr(pr[i].r,t) ;if(l)
{
a=l-pr[i].r;
printf("pos: %d\n",a);
pr[i].r[a]=pr[m].l;
}}}}}
printf("Eliminate common expression\n");
for(i=0;i<z;i++)
{
printf("%c\t=",pr[i].l);
printf("%s\n",pr[i].r);
}
for(i=0;i<z;i++)
{
for(j=i+1;j<z;j++)
{
q=strcmp(pr[i].r,pr[j].r);
if((pr[i].l==pr[j].l)&&!q)
{
pr[i].l='\0';
}
}
}
```

```
printf("Optimized code\n");
for(i=0;i<z;i++)
{
if(pr[i].l!='\0'){
printf("%c=",pr[i].l);
printf("%s\n",pr[i].r);
}
}
}
```

OUTPUT:



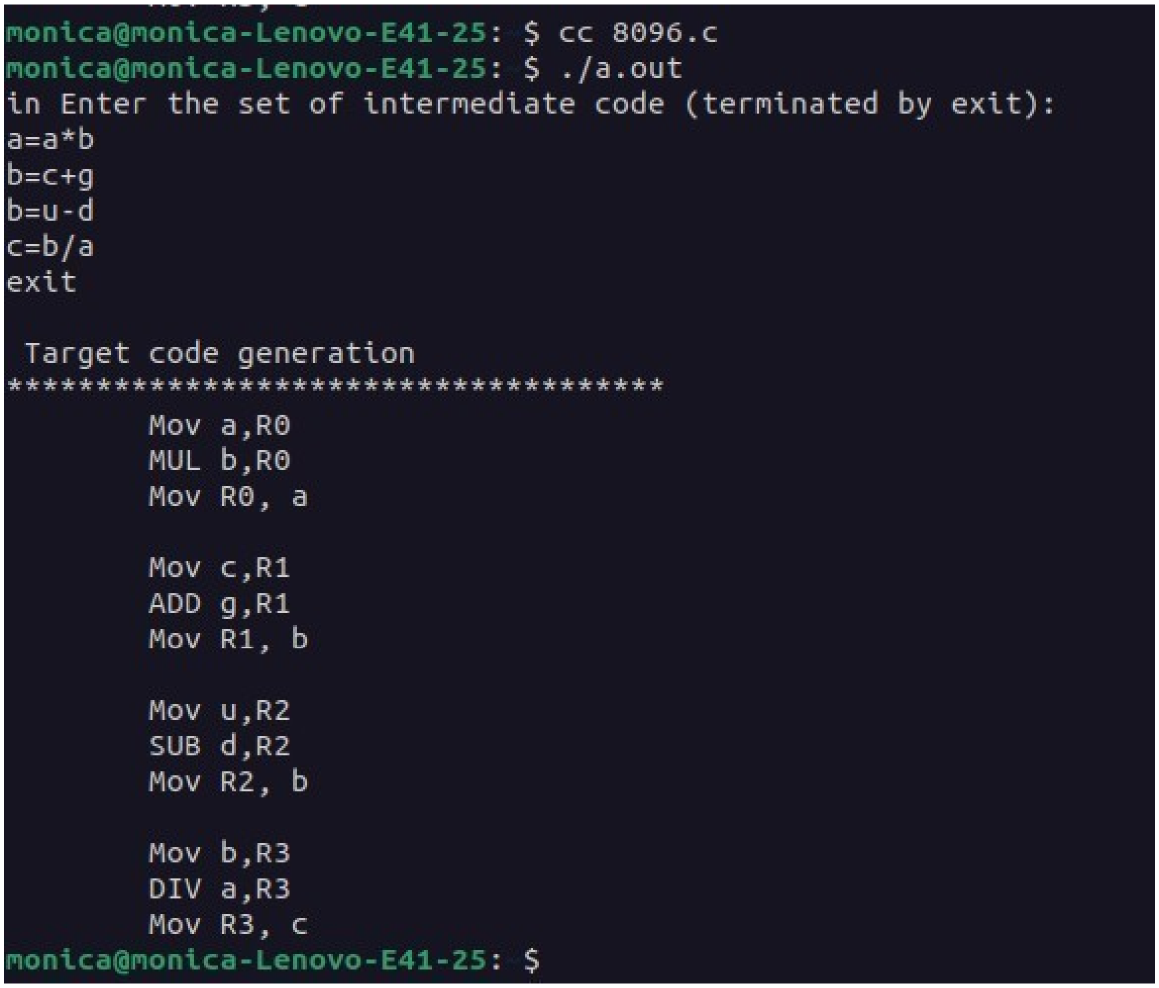
```
monica@monica-Lenovo-E41-25: $ cc code-opt.c
code-opt.c: In function 'main':
code-opt.c:22:10: warning: format '%s' expects argument of type 'char *', but argument 2 has type 'char (*)[20]' [-Wformat=]
   22 | scanf(" %s",&op[i].r);
      |          ~^  ~~~~~~~~
      |           |  |
      |           |  char (*)[20]
      |           char *
monica@monica-Lenovo-E41-25: $ ./a.out
Enter the Number of Values:5
left: a
right: x+y
left: b
right: a*z
left: c
right: a*z
left: d
right: c*x
left: e
right: b+d
Intermediate Code
a=x+y
b=a*z
c=a*z
d=c*x
e=b+d

After Dead Code Elimination
a       =x+y
a       =x+y
b       =a*z
c       =a*z
d       =c*x
e       =b+d
pos: 0
pos: 0
pos: 0
Eliminate Common Expression
a       =x+y
a       =x+y
b       =a*z
b       =a*z
d       =b*x
e       =b+d
Optimized Code
a=x+y
b=a*z
d=b*x
e=b+d
monica@monica-Lenovo-E41-25: $
```

PROGRAM:

```
#include <stdio.h>
#include <stdio.h>
#include <string.h>
void main()
{
char icode[10][30], str[20], opr[10];
int i = 0;
printf("in Enter the set of intermediate code (terminated by exit):\n");
do{
scanf("%s", icode[i]);
}while (strcmp(icode[i++], "exit") !=0);
printf("\n Target code generation");
printf("\n***********************************");
i = 0;
do {
strcpy(str, icode[i]);
switch (str[3])
{
case '+':
strcpy(opr, "ADD ");
break;
case '-':
strcpy(opr, "SUB ");
break;
case '*':
strcpy(opr, "MUL ");break;
case '/':
strcpy(opr, "DIV ");
break;
}
printf("\n\tMov %c,R%d", str[2], i);
printf("\n\t%s%c,R%d", opr, str[4], i);
printf("\n\tMov R%d, %c \n", i, str[0]);
}while (strcmp(icode[++i], "exit") != 0);
}
```

OUTPUT:



```
monica@monica-Lenovo-E41-25: $ cc 8096.c
monica@monica-Lenovo-E41-25: $ ./a.out
in Enter the set of intermediate code (terminated by exit):
a=a*b
b=c+g
b=u-d
c=b/a
exit

 Target code generation
*************************************
        Mov a,R0
        MUL b,R0
        Mov R0, a

        Mov c,R1
        ADD g,R1
        Mov R1, b

        Mov u,R2
        SUB d,R2
        Mov R2, b

        Mov b,R3
        DIV a,R3
        Mov R3, c
monica@monica-Lenovo-E41-25: $
```