UE20CS353-CD ASSIGNMENT-2

NAME	SRN	CLASS & SECTION
VIJAY J	PES2UG20CS815	6 - J

ABSTRACT SYNTAX TREE(AST):

lexer.l

Open ▾ lexer.l ~/PESU/6th Sem/CD/Assignment/Assignment-2/AST Save

```
%{
//NAME: Vijay J
//SRN: PES2UG20CS815
//SECTION: J
    #define YYSTYPE char*
    #include <unistd.h>
    #include "y.tab.h"
    #include <stdio.h>
    extern void yyerror(const char *); // declare the error handling function
%}

/* Regular definitions */
digit   [0-9]
letter  [a-zA-Z]
id      {letter}({letter}|{digit})*
digits  {digit}+
opFraction      (\.{digits})?
opExponent      ([Ee][+-]?{digits})?
number  {digits}{opFraction}{opExponent}
%option yylineno

%%
\/\/(.*) ; // ignore comments
[\t\n] ; // ignore whitespaces
"("             {return *yytext;}
")"             {return *yytext;}
"."             {return *yytext;}
","             {return *yytext;}
"*"             {return *yytext;}
"+"             {return *yytext;}
";"             {return *yytext;}
"-"             {return *yytext;}
"/"             {return *yytext;}
"="             {return *yytext;}
">"             {return *yytext;}
"<"             {return *yytext;}
"if"            {return IF;}
"else"          {return ELSE;}
"{"                             {return *yytext;}
"}"                             {return *yytext;}
{number}        {
                        yylval = strdup(yytext);  //stores the value of the number to be used later for symbol table insertion
                        return T_NUM;
                }
{id}            {
                                        yylval = strdup(yytext); //stores the identifier to be used later for symbol table insertion
                                        return T_ID;
                                }
.               {} // anything else ⇒ ignore
%%
```

parser.y

Open | *parser.y | ~/PESU/6th Sem/CD/Assignment/Assignment-2/AST | Save

```
%{
//NAME: Vijay J
//SRN: PES2UG20CS815
//SECTION: J
        #include "abstract_syntax_tree.c"
        #include <stdio.h>
        #include <stdlib.h>
        #include <string.h>
        void yyerror(char*
s);
error handling function
        int
yylex();
declare the function performing lexical analysis
        extern int
yylineno;
track the line number
%}

%union
union to allow nodes to store return different datatypes
{
        char* text;
        expression_node* exp_node;
}



%token <text> T_ID T_NUM IF ELSE

%type <text> RELOP
```

Open | *parser.y | ~/PESU/6th Sem/CD/Assignment/Assignment-2/AST | Save

```
%type <exp_node> E T F START ASSGN S C SEQ

/* specify start symbol */
%start START


%%
START : SEQ {
        display_exp_tree($1);
        printf("\nValid syntax\n");
        YYACCEPT;
};

SEQ : S SEQ { $$ = init_exp_node("seq", $1, $2,  NULL);}
        | S  {$$ = $1;}
;

S : IF '(' C ')' '{' SEQ '}' {
                $$ = init_exp_node(strdup("if"), $3, $6, NULL);
        }
        | IF '(' C ')' '{' SEQ '}' ELSE '{' SEQ '}' {
                $$ = init_exp_node(strdup("if-else"), $3, $6, $10);
        }
        | ASSGN { $$ = $1;}
;

C : F RELOP F {
        $$=init_exp_node(strdup($2), $1, $3, NULL);
}
;

```

Open | *parser.y | ~/PESU/6th Sem/CD/Assignment/Assignment-2/AST | Save

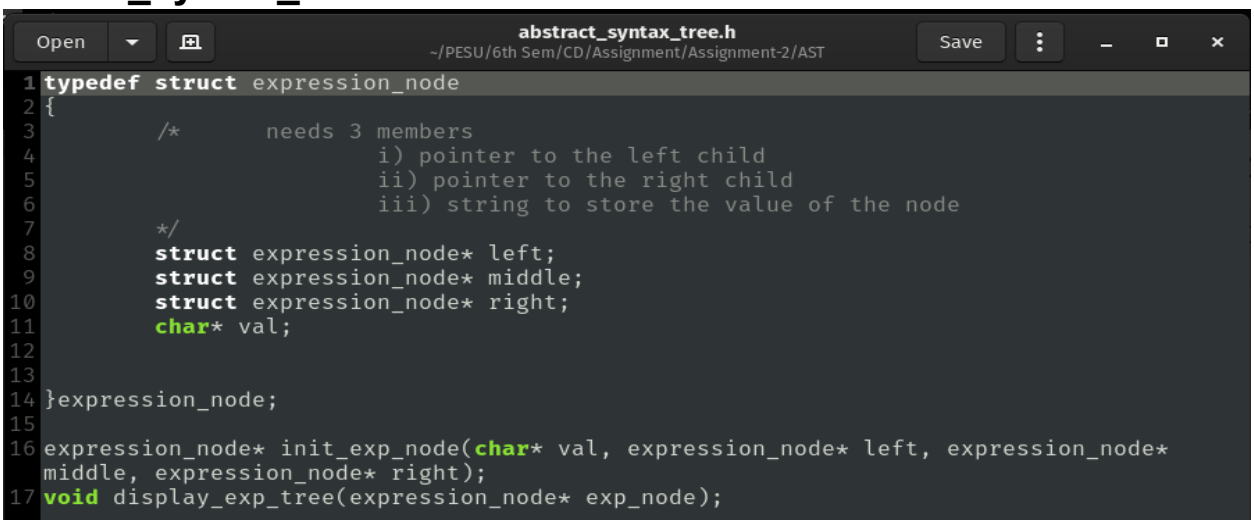
```
RELOP : '<' { $$ = "<";}
          | '>' { $$ = ">";}
          | '>''=' { $$ = "≥";}
          | '<''=' { $$ = "≤";}
          | '=''=' { $$ = "==";}
          | '!''=' { $$ = "≠";}
;



/* Grammar for assignment */
ASSGN : T_ID '=' E ';'  {
                                                $$ =
init_exp_node(strdup("=") ,init_exp_node(strdup($1) ,NULL ,NULL ,NULL) ,$3 ,NULL);
                                               }
        ;

/* Expression Grammar */
E : E '+' T     {
                                               $$ = init_exp_node(strdup("+") ,$1 ,
$3 ,NULL);
                                            }
        | E '-' T       {
                                               $$ = init_exp_node(strdup("-") ,$1 ,
$3 ,NULL);
                                            }
        | T     { $$ = $1; }
        ;


T : T '*' F     {
```

Open | *parser.y | ~/PESU/6th Sem/CD/Assignment/Assignment-2/AST | Save

```
                                               $$ = init_exp_node(strdup("*") ,$1 ,
$3 ,NULL);
                                            }
        | T '/' F       {
                                               $$ = init_exp_node(strdup("/") ,$1 ,
$3 ,NULL);
                                            }
        | F  { $$ = $1; }
        ;

F : '(' E ')' { $$ = $2; }
        | T_ID  {
                                    $$ = init_exp_node(strdup($1) ,NULL ,NULL ,NULL);
                                }
        | T_NUM         {
                                               $$ =
init_exp_node(strdup($1) ,NULL ,NULL ,NULL);
                                               }
        ;

%%


/* error handling function */
void yyerror(char* s)
{
        printf("Error :%s at %d \n",s,yylineno);
}

int yywrap() {
    return 1;
```

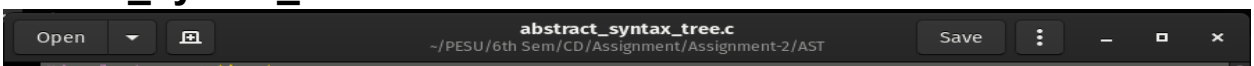
```
}

/* main function - calls the yyparse() function which will in turn drive yylex() as well */
int main(int argc, char* argv[])
{
        printf("Preorder:\n");
        yyparse();
        return 0;
}
```

abstract_syntax_tree.h




```
typedef struct expression_node
{
        /*      needs 3 members
                        i) pointer to the left child
                        ii) pointer to the right child
                        iii) string to store the value of the node
        */
        struct expression_node* left;
        struct expression_node* middle;
        struct expression_node* right;
        char* val;


}expression_node;

expression_node* init_exp_node(char* val, expression_node* left, expression_node* middle, expression_node* right);
void display_exp_tree(expression_node* exp_node);
```

abstract_syntax_tree.c




```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "abstract_syntax_tree.h"

expression_node* init_exp_node(char* val, expression_node* left, expression_node* middle, expression_node* right)
{
        // function to allocate memory for an AST node and set the left and right children of the nodes
        expression_node* node = (expression_node*)malloc(sizeof(expression_node));
        node->left = left;
        node->middle = middle;
        node->right = right;
        node->val = val;
}

void helper(expression_node* exp_node, int first)
{
        if(exp_node != NULL) {
                if(first) {
                        printf("%s",exp_node->val);
                        first = 0;
                }
                else {
                        printf(",%s", exp_node->val);
                }
                helper(exp_node->left, first);
                helper(exp_node->middle, first);
                helper(exp_node->right, first);
        }
}
```

```c
void display_exp_tree(expression_node* exp_node)
{
        // traversing the AST in preorder and displaying the nodes
        int first = 1;
        helper(exp_node, first);
}
```

run.sh

Open | run.sh | ~/PESU/6th Sem/CD/Assignment/Assignment-2/AST | Save

```
#!/bin/bash

lex lexer.l
yacc -d parser.y
gcc -g y.tab.c lex.yy.c

rm lex.yy.c
rm y.tab.c
rm y.tab.h

```

input code and output screenshot
test_input1.c

Open | test_input1.c | ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815 | Save

```
if(a > b)
{
   a = a + 1;
   b = b - 1;
}
```

Output

~/P/6/C/A/A/P/AST

```
yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 2ms
λ ./run.sh

yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 384ms
λ ./a.out < test_input1.c
Preorder:
if,>,a,b,seq,=,a,+,a,1,=,b,-,b,1
Valid syntax
```

test_input2.c

Open — test_input2.c — ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815 — Save

```
if(a > b)
{
 a = a + 1;
 b = b - 1;
}
else
{
 a = a - 1;
 b = b - 1;
}
```

Output

~/P/6/C/A/A/P/AST

```
yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 4ms
λ ./run.sh

yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 376ms
λ ./a.out < test_input2.c
Preorder:
if-else,>,a,b,seq,=,a,+,a,1,=,b,-,b,1,seq,=,a,-,a,1,=,b,-,b,1
Valid syntax
```

test_input3.c

Open — test_input3.c — ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815 — Save

```
if(a > b)
{
 a = a + 1;
 b = b - 1;
}
else
{
 a = a - 1;
 b = b - 1;
if (b < 0)
{
 b= b + 1;
}
else
{
 b = 0;
}
}
```

Output

~/P/6/C/A/A/P/AST

```
yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 2ms
λ ./run.sh

yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/AST via C v12.2.1-gcc took 362ms
λ ./a.out < test_input3.c
Preorder:
if-else,>,a,b,seq,=,a,+,a,1,=,b,-,b,1,seq,=,a,-,a,1,seq,=,b,-,b,1,if-else,<,b,0,=,b,+,b,1,=,b,0
Valid syntax
```

INTERMEDIATE CODE GENERATION(ICG):

lexer.l

Open | lexer.l — ~/PESU/6th Sem/CD/Assignment/Assignment-2/ICG | Save

```
%{
//NAME: Vijay J
//SRN: PES2UG20CS815
//SECTION: J
    #define YYSTYPE char*
    #include <unistd.h>
    #include "y.tab.h"
    #include <stdio.h>
    extern void yyerror(const char *); // declare the error handling function
%}

/* Regular definitions */
digit   [0-9]
letter  [a-zA-Z]
id      {letter}({letter}|{digit})*
digits  {digit}+
opFraction      (\.{digits})?
opExponent      ([Ee][+-]?{digits})?
number  {digits}{opFraction}{opExponent}
%option yylineno

%%
\/\/(.*) ; // ignore comments
[\t\n] ; // ignore whitespaces
"≤"             {return LTEQ;}
"≥"             {return GTEQ;}
"═"             {return EQQ;}
"≠"             {return NEQ;}
"{"             {return OC;}
"}"             {return CC;}
"("             {return *yytext;}
")"             {return *yytext;}
```

```
"."             {return *yytext;}
","             {return *yytext;}
"*"             {return *yytext;}
"+"             {return *yytext;}
";"             {return *yytext;}
"-"             {return *yytext;}
"/"             {return *yytext;}
"="             {return *yytext;}
">"             {return GT;}
"<"             {return LT;}
{number}        {
                        yylval = strdup(yytext);  //stores the value of the number to be used later for symbol table insertion
                        return T_NUM;
                }
"if"            {return T_IF;}
"else"          {return T_ELSE;}
{id}            {
                                        yylval = strdup(yytext); //stores the identifier to be used later for symbol table insertion
                                        return T_ID;
                                }
.               {} // anything else ⇒ ignore
%%
```

parser.y

Open | parser.y — ~/PESU/6th Sem/CD/Assignment/Assignment-2/ICG | Save

```
%{
//NAME: Vijay J
//SRN: PES2UG20CS815
//SECTION: J
        #include "quad_generation.c"
        #include <stdio.h>
        #include <stdlib.h>
        #include <string.h>

        #define YYSTYPE char*

        void yyerror(char*
s);
error handling function
        int
yylex();
declare the function performing lexical analysis
        extern int
yylineno;
track the line number

        FILE* icg_quad_file;
        int temp_no = 1;
        int label_no=1;
%}


%token T_ID T_NUM T_IF T_ELSE GTEQ LTEQ EQQ NEQ GT LT OC CC

/* specify start symbol */
%start START
```

Open | parser.y — ~/PESU/6th Sem/CD/Assignment/Assignment-2/ICG | Save

```
%nonassoc T_IF
%nonassoc T_ELSE

%%
START : S {
                printf("------------------------------------------------------------\n");
                printf("Valid syntax\n");
                YYACCEPT;
        };



/* Grammar for assignment */
ASSGN : T_ID '=' E      {

                                quad_code_gen($1, $3, "=", " ");

                        }
        ;

/* Expression Grammar */
E : E '+' T     {

                        $$ = new_temp();
                        quad_code_gen($$, $1, "+", $3);
                }
        | E '-' T       {

                                $$ = new_temp();
                                quad_code_gen($$, $1, "-", $3);
                        }
```

parser.y
~/PESU/6th Sem/CD/Assignment/Assignment-2/ICG

```
        | T
        ;


T : T '*' F     {
                        $$ = new_temp();
                        quad_code_gen($$, $1, "*", $3);
                }
        | T '/' F       {


                                $$ = new_temp();
                                quad_code_gen($$, $1, "/", $3);
                        }
        | F
        ;

F : '(' E ')'   {

                        $$=strdup($2);
                }
        | T_ID          {
                                        $$=strdup($1);
                                }
        | T_NUM         {
                                $$=strdup($1);
                        }
        ;

S : T_IF '('C')' OC S CC {quad_code_gen($3,"","Label","");} S
        | T_IF '('C')' OC S CC {
```

parser.y
~/PESU/6th Sem/CD/Assignment/Assignment-2/ICG

```
                        $2 = new_label();
                                quad_code_gen($2,"","goto","");
                                quad_code_gen($3,"","Label","");} T_ELSE OC S CC
{quad_code_gen($2,"","Label","");}S
        | ASSGN ';' S
        |'{'S'}'
        |
        ;

C : E rel E  {  $$ = new_temp();
                quad_code_gen($$, $1, $2, $3);
                $1 = new_label();
                quad_code_gen($1,$$,"if","");
                $$ = new_label();
                quad_code_gen($$,"","goto","");
                quad_code_gen($1,"","Label","");

                };

rel :  GT {strcpy($$,">");}
     | LT {strcpy($$,"<");}
     | LTEQ {strcpy($$,"≤");}
     | GTEQ {strcpy($$,"≥");}
     | EQQ {strcpy($$,"==");}
     | NEQ {strcpy($$,"≠");}
     ;

%%


/* error handling function */
```

```c
void yyerror(char* s)
{
        printf("Error :%s at %d \n",s,yylineno);
}

int yywrap() {
    return 1;
}

/* main function - calls the yyparse() function which will in turn drive yylex() as well */
int main(int argc, char* argv[])
{

        printf("Generated Intermediate Code \n");
        printf("-------------------------------------------------------------\n");
        printf("| %-10s | %-10s | %-10s | %-10s |\n", "op", "arg1", "arg2", "result");
        printf("-------------------------------------------------------------\n");
        yyparse();
        return 0;
}
```

quad_generation.h

quad_generation.h
~/PESU/6th Sem/CD/Assignment/Assignment-2/ICG

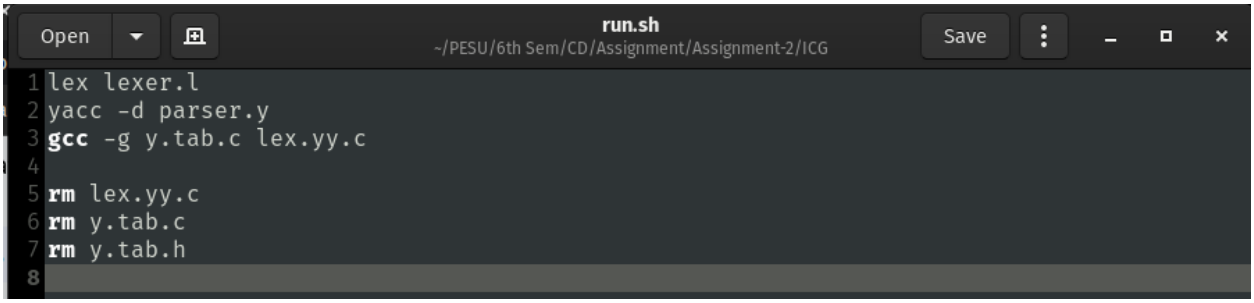
```c
extern FILE* icg_quad_file;             //pointer to the output file
extern int temp_no;                     //variable to keep track of current temporary count
extern int label_no;

void quad_code_gen(char* a, char* b, char* op, char* c);
char* new_temp();
```

quad_generation.c

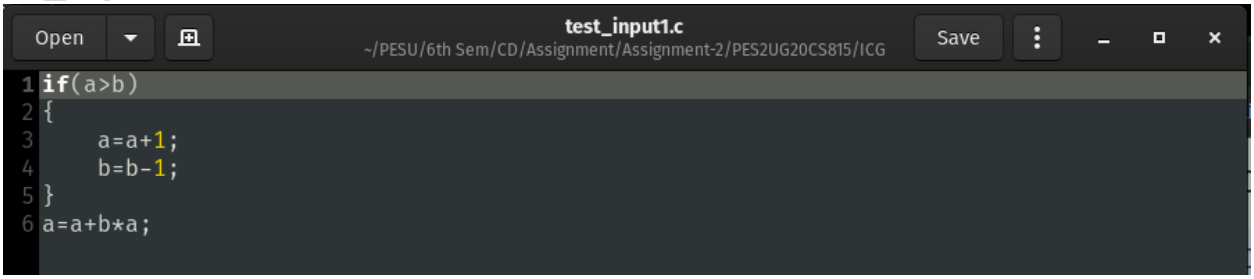
quad_generation.c
~/PESU/6th Sem/CD/Assignment/Assignment-2/ICG

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "quad_generation.h"


void quad_code_gen(char* a, char* b, char* op, char* c)
{
        printf("| %-10s | %-10s | %-10s | %-10s |\n",op,b,c,a);
}

char* new_temp()
{
                char* tempNew = (char*)malloc(sizeof(char)*4);
                sprintf(tempNew,"t%d",temp_no);
                temp_no++;
                return tempNew;
}

char* new_label()
{
                char* label = (char*)malloc(sizeof(char)*4);
                sprintf(label,"L%d",label_no);
                label_no++;
                return label;
}
```

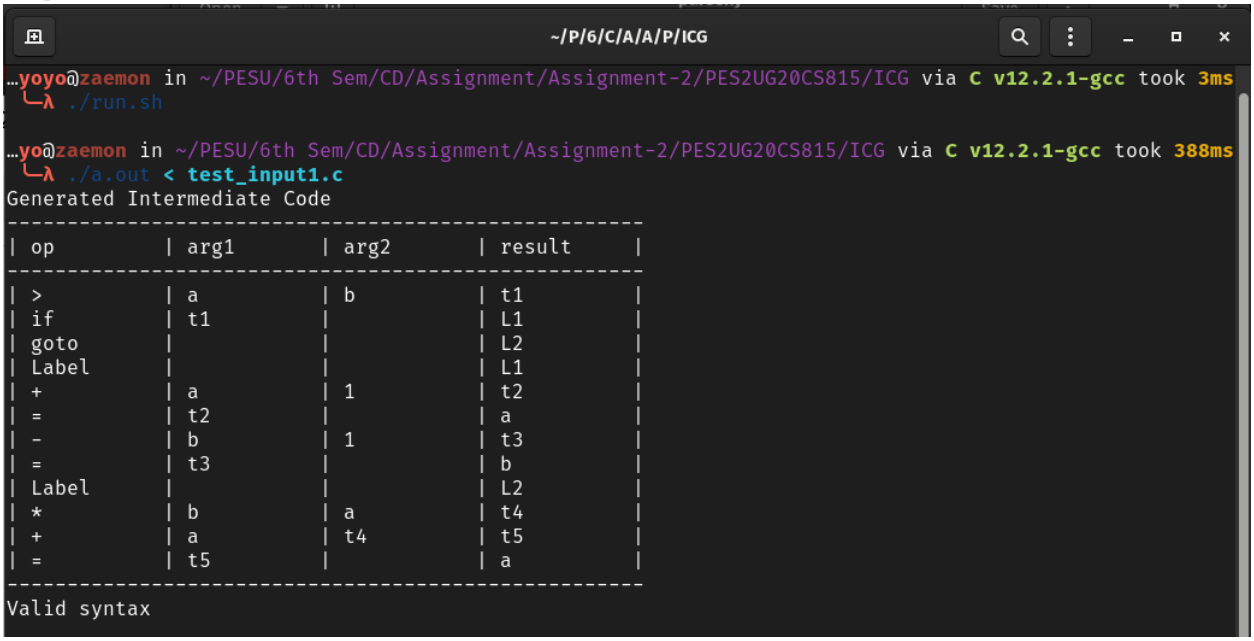
run.sh



input code and output screenshot
test_input1.c



Output



test_input2.c

Open | test_input2.c | ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/ICG | Save

```
1  if(a > b)
2  {
3    a = a + 1;
4    b = b - 1;
5  }
6  else
7  {
8    a = a - 1;
9    b = b - 1;
10 }
```

Output

~/P/6/C/A/A/P/ICG

```
yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/ICG via C v12.2.1-gcc took 51ms
λ ./run.sh

yoyo@zaemon in ~/PESU/6th Sem/CD/Assignment/Assignment-2/PES2UG20CS815/ICG via C v12.2.1-gcc took 1s
λ ./a.out < test_input2.c
Generated Intermediate Code
-----------------------------------------------------
| op          | arg1        | arg2        | result     |
-----------------------------------------------------
| >           | a           | b           | t1         |
| if          | t1          |             | L1         |
| goto        |             |             | L2         |
| Label       |             |             | L1         |
| +           | a           | 1           | t2         |
| =           | t2          |             | a          |
| -           | b           | 1           | t3         |
| =           | t3          |             | b          |
| goto        |             |             | L3         |
| Label       |             |             | L2         |
| -           | a           | 1           | t4         |
| =           | t4          |             | a          |
| -           | b           | 1           | t5         |
| =           | t5          |             | b          |
| Label       |             |             | L3         |
-----------------------------------------------------
Valid syntax
```