Project: Generate P-Code instructions for *expressions*. Implement semantics for the shaded rules in the table below. Implementation notes are written immediately below the rule. Write your p-code instructions to a file having the same prefix as the file containing the Subset Pascal Program but having the suffix "**.trc**"

Compile test programs in Table 2 and inspect the P-Code produced to validate that your Subset Pascal Compiler functions properly.

PCode Generation:

File	Description
PCode.h	Defines class PCode for creating p-code instructions
PCode.cpp	Implements class PCode
Node.h	Defines class Node for creating expression trees
Node.cpp	Implements class Node
Exp.h	Defines class Exp for printing expression trees
Exp.cpp	Implements class Exp

		Rule	
No.	**LHS**		**RHS**
37	*expression*	→	*simple_expression*
38	*expression*	→	*simple_expression* **=** *simple_expression*
39	*expression*	→	*simple_expression* **<>** *simple_expression*
40	*expression*	→	*simple_expression* **<** *simple_expression*
41	*expression*	→	*simple_expression* **<=** *simple_expression*
42	*expression*	→	*simple_expression* **>** *simple_expression*
43	*expression*	→	*simple_expression* **>=** *simple_expression*

Table 1. Subset Pascal Grammar (continued)

Test Program	**P-Code**
37 *expression* → *simple_expression*	
38 *expression* → *simple_expression* **=** *simple_expression*	
39 *expression* → *simple_expression* **<>** *simple_expression*	
40 *expression* → *simple_expression* **<** *simple_expression*	
41 *expression* → *simple_expression* **<=** *simple_expression*	
42 *expression* → *simple_expression* **>** *simple_expression*	
43 *expression* → *simple_expression* **>=** *simple_expression*	
program e00;	lvb 0 6
var a,b,c:boolean;	lvb 0 5
begin{e00}	lvb 0 6
a:=b;	equ b
a:=a=b;	lvb 0 5
a:=a<>b;	lvb 0 6
a:=a<b;	neq b
a:=a<=b;	lvb 0 5
a:=a>b;	lvb 0 6
a:=a>=b	grt b
end{e00}.	lvb 0 5

	lvb 0 6 geq b lvb 0 5 lvb 0 6 les b lvb 0 5 lvb 0 6 leq b

Table 2. Sample programs that exercise *expression* productions and corresponding P-Code fragments

Test Program	**P-Code**
37 *expression* → *simple_expression*	
38 *expression* → *simple_expression* **=** *simple_expression*	
39 *expression* → *simple_expression* **<>** *simple_expression*	
40 *expression* → *simple_expression* **<** *simple_expression*	
41 *expression* → *simple_expression* **<=** *simple_expression*	
42 *expression* → *simple_expression* **>** *simple_expression*	
43 *expression* → *simple_expression* **>=** *simple_expression*	

program e01;
 var a,b,c:char;
begin{e01}
 a:=b;
 a:=a=b;
 a:=a<>b;
 a:=a<b;
 a:=a<=b;
 a:=a>b;
 a:=a>=b
end{e01}.
	lvc 0 6
	lvc 0 5
	lvc 0 6
	equ c
	lvc 0 5
	lvc 0 6
	neq c
	lvc 0 5
	lvc 0 6
	grt c
	lvc 0 5
	lvc 0 6
	geq c
	lvc 0 5
	lvc 0 6
	les c
	lvc 0 5
	lvc 0 6
	leq c

Table 2. Sample programs that exercise *expression* productions and corresponding P-Code fragments (continued)

Test Program	**P-Code**
37 *expression* → *simple_expression*	
38 *expression* → *simple_expression* **=** *simple_expression*	
39 *expression* → *simple_expression* **<>** *simple_expression*	
40 *expression* → *simple_expression* **<** *simple_expression*	
41 *expression* → *simple_expression* **<=** *simple_expression*	
42 *expression* → *simple_expression* **>** *simple_expression*	
43 *expression* → *simple_expression* **>=** *simple_expression*	

program e02;
 var a,b,c:integer;
begin{e02}
 a:=b;
 a:=a=b;
 a:=a<>b;
 a:=a<b;
 a:=a<=b;
 a:=a>b;
 a:=a>=b
end{e02}.
	lvi 0 6
	lvi 0 5
	lvi 0 6
	equ i
	lvi 0 5
	lvi 0 6
	neq i
	lvi 0 5
	lvi 0 6
	grt i
	lvi 0 5
	lvi 0 6
	geq i
	lvi 0 5
	lvi 0 6
	les i
	lvi 0 5
	lvi 0 6
	leq i

Table 2. Sample programs that exercise *expression* productions and corresponding P-Code fragments (continued)

Test Program	**P-Code**
37 *expression* → *simple_expression*	
38 *expression* → *simple_expression* **=** *simple_expression*	
39 *expression* → *simple_expression* **<>** *simple_expression*	
40 *expression* → *simple_expression* **<** *simple_expression*	
41 *expression* → *simple_expression* **<=** *simple_expression*	
42 *expression* → *simple_expression* **>** *simple_expression*	
43 *expression* → *simple_expression* **>=** *simple_expression*	

program e03;
 var a,b,c:real;
begin{e03}
 a:=b;
 a:=a=b;
 a:=a<>b;
 a:=a<b;
 a:=a<=b;
 a:=a>b;
 a:=a>=b
end{e03}.
	lvr 0 6
	lvr 0 5
	lvr 0 6
	equ r
	lvr 0 5
	lvr 0 6
	neq r
	lvr 0 5
	lvr 0 6
	grt r
	lvr 0 5
	lvr 0 6
	geq r
	lvr 0 5
	lvr 0 6
	les r
	lvr 0 5
	lvr 0 6
	leq r

Table 2. Sample programs that exercise *expression* productions and corresponding P-Code fragments (continued)