Project: Generate P-Code instructions for *expression_list*, *statement*, *statement_list*, *optional_statements*, and *compound_statement*. Implement semantics for the shaded rules in the table below. Write your p-code instructions to a file having the same prefix as the file containing the Subset Pascal Program but having the suffix "**.pcd**"

Compile test programs in Tables 2, 3, and 4 and inspect the shaded P-Code produced to validate that your Subset Pascal Compiler functions properly.

		Rule	
No.	**LHS**		**RHS**
1	*program*	→	*program_head program_declarations program_body*
2	*program_head*	→	**program id** *program_parameters* **;**
3	*program_declarations*	→	*variable_declarations subprogram_definitions*
4	*program_body*	→	*compound_statement* **.**
5	*program_parameters*	→	∈
6	*program_parameters*	→	**(** *program_parameter_list* **)**
7	*program_parameter_list*	→	*identifier_list*
8	*identifier_list*	→	**id**
9	*identifier_list*	→	*identifier_list* **, id**
10	*variable_declarations*	→	∈
11	*variable_declarations*	→	*variable_declarations* **var** *identifier_list* **:** *type* **;**
12	*type*	→	*standard_type*
13	*type*	→	**array [intlit .. intlit] of** *standard_type*
14	*standard_type*	→	**id**
15	*subprogram_definitions*	→	∈
16	*subprogram_definitions*	→	*subprogram_definitions subprogram_definition* **;**
17	*subprogram_definition*	→	*subprogram_head variable_declarations subprogram_body*
18	*subprogram_head*	→	*function_prolog subprogram_parameters* **:** *standard_type* **;**
19	*subprogram_head*	→	*procedure_prolog subprogram_parameters* **;**
20	*function_prolog*	→	**function id**
21	*procedure_prolog*	→	**procedure id**
22	*subprogram_body*	→	*compound_statement*
23	*subprogram_parameters*	→	∈
24	*subprogram_parameters*	→	**(** *parameter_list* **)**

Table 1. Subset Pascal Grammar

No.	LHS		Rule RHS
25	*parameter_list*	→	*identifier_list* **:** *type*
26	*parameter_list*	→	*parameter_list* **;** *identifier_list* **:** *type*
27	*compound_statement*	→	**begin** *optional_statements* **end**
28	*optional_statements*	→	∈
29	*optional_statements*	→	*statement_list*
30	*statement_list*	→	*statement*
31	*statement_list*	→	*statement_list* **;** *statement*
32	*statement*	→	*variable* **:=** *expression*
33	*statement*	→	*procedure_statement*
34	*statement*	→	*compound_statement*
35	*statement*	→	**if** *expression* **then** *statement* **else** *statement*
36	*statement*	→	**while** *expression* **do** *statement*
37	*variable*	→	**id**
38	*variable*	→	**id [** *expression* **]**
39	*procedure_statement*	→	**id**
40	*procedure_statement*	→	**id (** *expression_list* **)**
41	*expression_list*	→	*expression*
42	*expression_list*	→	*expression_list* **,** *expression*
43	*expression*	→	*simple_expression*
44	*expression*	→	*simple_expression* = *simple_expression*
45	*expression*	→	*simple_expression* <> *simple_expression*
46	*expression*	→	*simple_expression* < *simple_expression*
47	*expression*	→	*simple_expression* <= *simple_expression*
48	*expression*	→	*simple_expression* > *simple_expression*
49	*expression*	→	*simple_expression* >= *simple_expression*
50	*simple_expression*	→	*term*
51	*simple_expression*	→	*+ term*
52	*simple_expression*	→	*- term*
53	*simple_expression*	→	*simple_expression* + *term*
54	*simple_expression*	→	*simple_expression* - *term*
55	*simple_expression*	→	*simple_expression* **or** *term*
56	*term*	→	*factor*
57	*term*	→	*term* ***** *factor*
58	*term*	→	*term* / *factor*
59	*term*	→	*term* **div** *factor*
60	*term*	→	*term* **mod** *factor*
61	*term*	→	*term* **and** *factor*

Table 1. Subset Pascal Grammar (continued)

			Rule
No.	**LHS**		**RHS**
62	*factor*	→	**id**
63	*factor*	→	**id [** *expression* **]**
64	*factor*	→	**id (** *expression_list* **)**
65	*factor*	→	**(** *expression* **)**
66	*factor*	→	**not** *factor*
67	*factor*	→	**intlit**
68	*factor*	→	**realit**
69	*factor*	→	**chrlit**

Table 1. Subset Pascal Grammar (continued)

Test Program			**P-Code**			
41	*expression_list*	→	*expression*			
42	*expression_list*	→	*expression_list* **,** *expression*			
program *el00***;**			L004	ent	sp	L005
var *a*,*b*:*integer***;**				ent	ep	L006
var *u*,*v*:*real***;**				lda	0	0
function *f***(***i*:*integer***;***r*:*real***):***real***;**				lvi	0	5
begin{f} *f***:=***i*+*r* **end{f};**				flt		
begin{el00}				lvr	0	6
*u***:=***f***(***a*+*b****6,***u*+*v****9.8)**				adr		
end{el00}.				sti	r	
				rtn	r	
			#define	L005	7	
			#define	L006	10	
			L001	ent	sp	L002
				ent	ep	L003
				lda	0	7
				mst	0	
				lvi	0	5
				lvi	0	6
				ldc	i	6
				mpi		
				adi		
				lvr	0	7
				lvr	0	8
				ldc	r	9.8
				mpr		
				adr		
				cup	2	L004
				sti	r	
				rtn	p	
			#define	L002	9	
			#define	L003	19	
				mst	0	
				cup	0	L001
				stp		

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

Test Program	**P-Code**
32 *statement* → *variable* **:=** *expression*	
program *s00***;** **var** *a*,*b*,*c*,*d***:***integer***;** **begin{s00}** *a***:=***b*+*c******d* **end{s00}.**	L001 ent sp L002 ent ep L003 lda 0 5 lvi 0 6 lvi 0 7 lvi 0 8 mpi adi sti i rtn p #define L002 9 #define L003 13 mst 0 cup 0 L001 stp

Table 3. Sample programs that exercise *statement*→*variable***:=***expression* production and corresponding P-Code fragments

Test Program	**P-Code**
33 *statement* → *procedure_statement*	
program *s01***;** **procedure** *p***; begin{p} end{p};** **begin{s01}** *p* **end{s01}.**	L004 ent sp L005 ent ep L006 rtn p #define L005 5 #define L006 5 L001 ent sp L002 ent ep L003 mst 0 cup 0 L004 rtn p #define L002 5 #define L003 10 mst 0 cup 0 L001 stp

Table 3. Sample programs that exercise *statement*→*procedure_statement* productions and corresponding P-Code fragments (continued)

Test Program	**P-Code**
33 *statement* → *procedure_statement*	

program *s02***;**
  var *k***:***integer***;**
  var *c***:***boolean***;**
  procedure *p***(***i***:***integer***;***b***:***boolean***);**
  begin{p} end{p};
begin{s02}
  *p***(***k***+5,not** *c***)**
end{s02}.
	`L004 ent sp L005`
	`ent ep L006`
	`rtn p`
	`#define L005 7`
	`#define L006 7`
	`L001 ent sp L002`
	`ent ep L003`
	`mst 0`
	`lvi 0 5`
	`ldc i 5`
	`adi`
	`lvb 0 6`
	`not`
	`cup 2 L004`
	`rtn p`
	`#define L002 7`
	`#define L003 14`
	`mst 0`
	`cup 0 L001`
	`stp`

Table 3. Sample programs that exercise *statement*→*procedure_statement* productions and corresponding P-Code fragments (continued)

Test Program	**P-Code**
34 *statement* → *compound_statement*	

program *s03***;**
  var *b***:***boolean***;**
  var *c***:***char***;**
  var *i***:***integer***;**
begin{s03}
  *b***:=not true;**
  *c***:='a';**
  *i***:=29**
end{s03}.
	`L001 ent sp L002`
	`ent ep L003`
	`lda 0 5`
	`ldc b 1`
	`not`
	`sti b`
	`lda 0 6`
	`ldc c 'a'`
	`sti c`
	`lda 0 7`
	`ldc i 29`
	`sti i`
	`rtn p`
	`#define L002 8`
	`#define L003 10`
	`mst 0`
	`cup 0 L001`
	`stp`

Table 3. Sample programs that exercise *statement* → *compound_statement* productions and corresponding P-Code fragments (continued)

Test Program	**P-Code**
35 *statement* → **if** *expression* **then** *statement* **else** *statement*	

```
program s04;
    var a,b,c:integer;
begin{s04}
    a:=1; b:=2; c:=3;
    if a<b then a:=c else b:=c;
    a:=29
end{s04}.
```

```
L001        ent         sp          L002
            ent         ep          L003
            lda         0           5
            ldc         i           1
            sti         i
            lda         0           6
            ldc         i           2
            sti         i
            lda         0           7
            ldc         i           3
            sti         i
            lvi         0           5
            lvi         0           6
            grt         i
            fjp                     L004
            lda         0           5
            lvi         0           7
            sti         i
            ujp                     L005
L004
            lda         0           6
            lvi         0           7
            sti         i
L005
            lda         0           5
            ldc         i           29
            sti         i
            rtn         p
#define     L002        8
#define     L003        x
            mst         0
            cup         0           L001
            stp
```

Table 3. Sample programs that exercise *statement*→**if** *expression* **then** *statement* **else** *statement* productions and corresponding P-Code fragments (continued)

Test Program	**P-Code**
36 *statement* → **while** *expression* **do** *statement*	

program *s05*;
var *a*,*c*:*integer*;
begin{s05}
c:=**10;**
while *c*>**0 do** *c*:=*c*-**1;**
a:=**29**
end{s05}.
	`L001 ent sp L002`
	`ent ep L003`
	`lda 0 6`
	`ldc i 10`
	`sti i`
	`L004`
	`lvi 0 6`
	`ldc i 0`
	`les i`
	`fjp L005`
	`lda 0 6`
	`lvi 0 6`
	`ldc i 1`
	`sbi`
	`sti i`
	`ujp L004`
	`L005`
	`lda 0 5`
	`ldc i 29`
	`sti i`
	`rtn p`
	`#define L002 7`
	`#define L003 10`
	`mst 0`
	`cup 0 L001`
	`stp`

Table 3. Sample programs that exercise *statement*→ **while** *expression* **do** *statement* productions and corresponding P-Code fragments (continued)

Test Program	**P-Code**
30 *statement_list* → *statement*	
31 *statement_list* → *statement_list* **;** *statement*	
28 *optional_statements* → ∈	
29 *optional_statements* → *statement_list*	
27 *compound_statement* → **begin** *optional_statements* **end**	
Refer to Table 3.	

Table 4. Sample programs that exercise *statement_list* → *statement* productions and corresponding P-Code fragments (continued)