Project: Implement standard procedures and functions as given in Tables 1 and 2. Generate P-Code instructions for rules 40 and 64. 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 Table 4, 5, and 6, and inspect the P-Code produced to validate that your Subset Pascal Compiler functions properly.

Standard Procedure	Description
procedure *WriteBoolean*(*b*:*boolean*,*w*:*integer*);	Procedure *WriteBoolean* prints the Boolean value of parameter *b* on the display in a field of *w* characters. The value is right-justified. The Boolean value is printed as either **false** or **true**.
procedure *WriteChar*(*c*:*char*,*w*:*integer*);	Procedure *WriteChar* prints the character value of parameter *c* on the display in a field of *w* characters. The value is right-justified.
procedure *WriteInteger*(*i*:*integer*,*w*:*integer*);	Procedure *WriteInteger* prints the integer value of parameter *i* on the display in a field of *w* characters. The value is right-justified.
procedure *WriteExponential*(*r*:*real*;*w*,*f*:*integer*);	Procedure *WriteExponential* prints the real value of parameter *r* on the display in a field of *w* characters. Real parameter *r* is formatted in scientific notation. Parameter *f* governs the number of fractional digits printed. The value is right-justified.
procedure *WriteFixed*(*r*:*real*;*w*,*f*:*integer*);	Procedure *WriteFixed* prints the real value of parameter *r* on the display in a field of *w* characters. Real parameter *r* is formatted in fixed notation. Parameter *f* governs the number of fractional digits printed. The value is right-justified.

Table 1. Standard Procedures

Standard Function	Description
function *abs*(*i*:*integer*):*integer*;	Function *abs* returns the magnitude of its integer parameter *i*.
function *abs*(*r*:*real*):*real*;	Function *abs* returns the magnitude of its real parameter *r*.
function *trunc*(*r*:*real*):*integer*;	Function *trunc* returns an integer that is the truncated portion of real parameter *r*.
function *round*(*r*:*real*):*integer*;	Function *round* returns the integer that is closest to real parameter *r*.
function *ord*(*b*:*boolean*):*integer*;	Function *ord* returns the integer value of Boolean parameter *b*. Zero is returned for **false** and one is returned for **true**.
function *ord*(*c*:*char*):*integer*;	Function *ord* returns the ASCII integer code of character parameter *c*. For example, integer value 97 is returned for the ASCII character 'a.'
function *chr*(*i*:*integer*):*char*;	Function *chr* returns the character for which the value of the integer parameter *i* is the ASCII code.
function *succ*(*b*:*boolean*):*boolean*;	Function *succ* returns **true** if the value of the input Boolean parameter *b* is **false**.

Table 2. Standard Functions

Standard Function	Description
function *succ***(***c***:***char***):***char***;**	Function *succ* returns the character whose ASCII code is one greater than parameter *c* (if it exists).
function *succ***(***i***:***integer***):***integer***;**	Function *succ* adds one to parameter *i* and returns the sum if it exists.
function *pred***(***b***:***boolean***):***boolean***;**	Function *pred* returns **false** if the value of the input Boolean parameter *b* is **true**.
function *pred***(***c***:***char***):***char***;**	Function *pred* returns the character whose ASCII code is one less than parameter *c* (if it exists).
function *pred***(***i***:***integer***):***integer***;**	Function *pred* subtracts one from parameter *i* and returns the difference if it exists.

Table 2. Standard Functions (continued)

		Rule	
No.	**LHS**		**RHS**
40	*procedure_statement*	→	**id (** *expression_list* **)**
64	*factor*	→	**id (** *expression_list* **)**

Table 3. Subset Pascal Grammar

Test Program	P-Code

program *example*;
  var *x*,*y*,*z*:*integer*;
  function *gcd*(*a*,*b*:*integer*):*integer*;
  begin{gcd}
  if *b*=**0 then** *gcd*:=*a* **else** *gcd*:=*gcd*(*b*,*a* **mod** *b*)
  end{gcd};
begin{example}
  x:=**105;** *y*:=**15;**
  WriteInteger(*gcd*(*x*,*y*),**5);**
  WriteLn
end{example}.
	`L004 ent sp L005`
	`ent ep L006`
	`lvi 0 6`
	`ldc i 0`
	`equ i`
	`fjp L007`
	`lda 0 0`
	`lvi 0 5`
	`sti i`
	`ujp L008`
	`L007`
	`lda 0 0`
	`mst 1`
	`lvi 0 6`
	`lvi 0 5`
	`lvi 0 6`
	`mod`
	`cup 2 L004`
	`sti i`
	`L008`
	`rtn i`
	`#define L005 7`
	`#define L006 16`
	`L001 ent sp L002`
	`ent ep L003`
	`lda 0 5`
	`ldc i 105`
	`sti i`
	`lda 0 6`
	`ldc i 15`
	`sti i`
	`mst 0`
	`lvi 0 5`
	`lvi 0 6`
	`cup 2 L004`
	`ldc i 5`
	`csp 0 wri`
	`csp wln`
	`rtn p`
	`#define L002 8`
	`#define L003 24`
	`mst 0`
	`cup 0 L001`
	`stp`

Table 4. Sample program containing standard procedures *WriteInteger* and *Writeln*

Test Program	**P-Code**			
program *csp00***;**	L001	ent	sp	L002
var *b***:***boolean***;**		ent	ep	L003
var *c***:***char***;**		lvb	0	5
var *i***:***integer***;**		ldc	i	6
var *r***:***real***;**		csp	0	wrb
begin{csp00}		lvc	0	6
*WriteBoolean***(***b***,6);**		ldc	i	2
*WriteChar***(***c***,2);**		csp	0	wrc
*WriteInteger***(***i***,5);**		lvi	0	7
*WriteExponential***(***r***,20);**		ldc	i	5
*WriteFixed***(***r***,20);**		csp	0	wri
Writeln		lvr	0	8
end{csp00}.		ldc	i	20
		csp	0	wre
		lvr	0	8
		ldc	i	20
		csp	0	wrf
		csp		wln
		rtn	p	
	#define	L002	9	
	#define	L003	11	
		mst	0	
		cup	0	L001
		stp		

Table 5. Sample programs that exercise standard procedures.

Test Program	**P-Code**			
program *csf00***;**	L001	ent	sp	L002
var *b***:***boolean***;**		ent	ep	L003
var *c***:***char***;**		lda	0	7
var *i***:***integer***;**		lvb	0	5
var *r***:***real***;**		ord		
begin{csf00}		sti	i	
*i***:=***ord***(***b***);**		lda	0	7
*i***:=***ord***(***c***);**		lvc	0	6
*c***:=***chr***(***i***);**		ord		
*i***:=***round***(***r***);**		sti	i	
*i***:=***trunc***(***r***);**		lda	0	6
*b***:=***succ***(***b***);**		lvi	0	7
*c***:=***succ***(***c***);**		chr		
*i***:=***succ***(***i***);**		sti	c	
*b***:=***pred***(***b***);**		lda	0	7
*c***:=***pred***(***c***);**		lvr	0	8
*i***:=***pred***(***i***)**		rnd		
end{csf00}.		sti	i	
		lda	0	7
		lvr	0	8
		trc		
		sti	i	

Table 6. Sample programs that exercise standard functions

Test Program	P-Code
	lda 0 5 lvb 0 5 inc sti b lda 0 6 lvc 0 6 inc sti c lda 0 7 lvi 0 7 inc sti i lda 0 5 lvb 0 5 dec sti b lda 0 6 lvc 0 6 dec sti c lda 0 7 lvi 0 7 dec sti i rtn p #define L002 9 #define L003 11 mst 0 cup 0 L001 stp

Table 6. Sample programs that exercise standard functions (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)