161	¡	225	Æ
162	¢	226	
163	£	227	ª
164	⁄	228	
165	¥	229	
166	ƒ	230	
167	§	231	
168	¤	232	Ł
169	'	233	Ø
170	“	234	Œ
171	«	235	º
172	‹	236	
173	›	237	
174	fi	238	
175	fl	239	
176		240	
177	–	241	æ
178	†	242	
179	‡	243	
180	·	244	
181		245	ı
182	¶	246	
183	•	247	
184	‚	248	ł
185	„	249	ø
186	”	250	œ
187	»	251	ß
188	…		
189	‰		
190			
191	¿		
192			
193	`		
194	´		
195	ˆ		
196	˜		
197	¯		
198	˘		
199	˙		
200	¨		
201			
202	°		
203	¸		
204			
205	˝		
206	˛		
207	ˇ		
208	—		

before they can be used (see *POSTSCRIPT Language Cookbook*). For a complete list of the characters and corresponding codes available in the standard POSTSCRIPT fonts, refer to the *POSTSCRIPT Language Reference Manual*.

Character codes may be directly used in two ways: they may be inserted into a string with a **put** operation or used directly in a string as an octal (base eight) number.

Putting Codes Into Strings

The following program uses **put** to generate a table of the characters whose standard codes are greater than 160. Note that some of the codes listed have no characters associated with them.

```
% -------- Variables & Procedures ---------
/Times-Roman findfont 10 scalefont setfont

/char 1 string def
/nstr 3 string def

/newline
 { currentpoint 11 sub
   exch pop LM
   exch moveto } def
(/prt-n        %stack: code
 {nstr cvs show} def

/prtchar      %stack: code
 { char  0
   3 –1 roll put
   char  show } def

/PrintCodeandChar                  %stack: code
 { dup prt-n
   (   ) show
   prtchar newline } def
% ---------- Begin Program ----------
/LM 72 def
LM 600 moveto
161 1 208 {PrintCodeandChar} for
```

```
/LM 144 def
LM 600 moveto
225 1 251 {PrintCodeandChar} for

showpage
```

The *prt-n* procedure defined above takes a number from the stack and prints it on the current page.

Prtchar takes a numeric code from the stack and prints the correspoding character. The procedure does this by putting the number into a one-character string and then printing the string. The first line

```
char  0
```

places the string and the index for the **put** on the stack. (Note that the only position in a one-character string is zero.) The next line

```
3 –1 roll put
```

brings the numeric code to the top of the stack and puts it into *char*. Finally, the procedure prints *char*, which now contains our character code.

The procedure *PrintCodeandChar* calls *prt-n*, prints three spaces, and then calls *prtchar*, thereby printing one line of our table.

```
/PrintCodeandChar              %stack: code
 { dup prt-n
   (   ) show
   prtchar newline } def
```

The program itself sets *LM*, our left margin, to *72*, moves to the top of the page, and then calls *PrintCodeandChar* for each number between *161* and *208*. It then resets the left margin to *144* and prints table entries for the numbers from *225* to *251*. The codes from *209* through *224* are skipped because they have no characters assigned to them in the standard encoding.