```
 1: <?xml version="1.0" encoding="iso-8859-1"?>
 2: <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 3:   <xsl:output encoding="iso-8859-1" indent="yes" method="xml" version="1.0"/>
 4:
 5:   <!-- pass the document's children to our converter -->
 6:   <xsl:template match="/">
 7:     <xsl:call-template name="converter">
 8:       <xsl:with-param name="nodes" select="child::node()"/>
 9:     </xsl:call-template>
10:   </xsl:template>
11:
12:   <!-- convert all attributes to child elements, leaving rest of document as-is -->
13:   <xsl:template name="converter">
14:     <xsl:param name="nodes"/>
15:
16:     <!-- iterate through each node in given nodeset-->
17:     <xsl:for-each select="$nodes">
18:
19:       <!-- act differently based on each node's type -->
20:       <xsl:choose>
21:
22:         <!-- else if current node's a comment, output it as-is -->
23:         <xsl:when test="self::comment()">
24:           <xsl:comment><xsl:value-of select="."/></xsl:comment>
25:         </xsl:when>
26:
27:         <!-- else if current node's a PI, output it as-is -->
28:         <xsl:when test="self::processing-instruction()">
29:           <xsl:processing-instruction name="{name()}"><xsl:value-of select="."/></xsl:processing-instruction>
30:         </xsl:when>
31:
32:         <!-- else if current node's text, output it as-is -->
33:         <xsl:when test="self::text()"><xsl:value-of select="."/></xsl:when>
34:
35:         <!-- else if current node's an element, output it as-is -->
36:         <xsl:otherwise>
37:           <xsl:element name="{name()}">
38:
39:             <!-- output each attribute as a child element -->
40:             <xsl:for-each select="attribute::*">
41:               <xsl:element name="{name()}"><xsl:value-of select="."/></xsl:element>
42:             </xsl:for-each>
43:
44:             <!-- now convert this node's children -->
45:             <xsl:call-template name="converter">
```

```
46:               <xsl:with-param name="nodes" select="child::node()"/>
47:             </xsl:call-template>
48:           </xsl:element>
49:         </xsl:otherwise>
50:
51:       </xsl:choose>
52:
53:     </xsl:for-each>
54:   </xsl:template>
55: </xsl:stylesheet>
```

```
 1: <?xml version="1.0" encoding="iso-8859-1"?>
 2: <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 3:   <xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="yes"/>
 4:
 5:   <!-- start recursive-descent processing at the root -->
 6:   <xsl:template match="/">
 7:     <xsl:apply-templates select="child::node()"/>
 8:   </xsl:template>
 9:
10:   <!-- output elements, with attributes converted to child elements -->
11:   <xsl:template match="*">
12:     <xsl:element name="{name()}">
13:       <!-- convert attributes to child elements -->
14:       <xsl:for-each select="attribute::*">
15:         <xsl:element name="{name()}"><xsl:value-of select="."/></xsl:element>
16:       </xsl:for-each>
17:       <!-- preserve namespace nodes -->
18:       <xsl:for-each select="namespace::*">
19:         <!-- not easy; best to wait for XSLT 2.0 -->
20:       </xsl:for-each>
21:       <xsl:apply-templates select="child::node()"/>
22:     </xsl:element>
23:   </xsl:template>
24:
25:   <!-- output text verbatim -->
26:   <xsl:template match="text()">
27:     <xsl:value-of select="."/>
28:   </xsl:template>
29:
30:   <!-- output PIs verbatim -->
31:   <xsl:template match="processing-instruction()">
32:     <xsl:processing-instruction name="{name()}"><xsl:value-of select="."/></xsl:processing-instruction>
33:   </xsl:template>
34:
35:   <!-- output comments verbatim -->
36:   <xsl:template match="comment()">
37:     <xsl:comment><xsl:value-of select="."/></xsl:comment>
38:   </xsl:template>
39:
40: </xsl:stylesheet>
```

```
 1: <?xml version="1.0" encoding="iso-8859-1"?>
 2: <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 3:   <xsl:output encoding="iso-8859-1" indent="yes" method="xml" version="1.0"/>
 4:
 5:   <!-- copy all nodes to output, but transform attributes to child elements -->
 6:   <xsl:template match="node()">
 7:     <xsl:copy>
 8:       <xsl:for-each select="attribute::*">
 9:         <xsl:element name="{name()}"><xsl:value-of select="."/></xsl:element>
10:       </xsl:for-each>
11:       <xsl:apply-templates select="child::node()"/>
12:     </xsl:copy>
13:   </xsl:template>
14:
15: </xsl:stylesheet>
```

```
    1: <?xml version="1.0" encoding="iso-8859-1"?>
    2:
    3: <xsl:stylesheet version="1.0" xmlns:xalan="http://xml.apache.org/xslt" xmlns:xsl="http://www.w3.org/1999/XSL/Transf
orm">
    4:
    5:     <!-- output pretty-printed results as XHTML 1.0 -->
    6:     <xsl:output doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN" doctype-system="http://www.w3.org/TR/xhtml1
/DTD/xhtml1-transitional.dtd" encoding="UTF-8" indent="yes" method="xml" xalan:indent-amount="4"/>
    7:
    8:     <xsl:template match="/">
    9:
   10:         <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
   11:
   12:             <head>
   13:                 <title>1</title>
   14:             </head>
   15:
   16:             <body>
   17:
   18:                 <!-- iterate over movie elements, sorted by title -->
   19:                 <xsl:for-each select="database/movies/movie">
   20:                     <xsl:sort select="title"/>
   21:
   22:                     <!-- output tiles of R movies in red, G movies in green, and others in black -->
   23:                     <xsl:choose>
   24:                         <xsl:when test="rating='R'">
   25:                             <font color="red">
   26:                                 <xsl:value-of select="title"/>
   27:                             </font>
   28:                         </xsl:when>
   29:                         <xsl:when test="rating='G'">
   30:                             <font color="green">
   31:                                 <xsl:value-of select="title"/>
   32:                             </font>
   33:                         </xsl:when>
   34:                         <xsl:otherwise>
   35:                             <xsl:value-of select="title"/>
   36:                         </xsl:otherwise>
   37:                     </xsl:choose>
   38:
   39:                     <br/>
   40:
   41:                 </xsl:for-each>
   42:
   43:             </body>
```

```
44:
45:         </html>
46:
47:     </xsl:template>
48: </xsl:stylesheet>
```

```
 1: <?xml version="1.0" encoding="iso-8859-1"?>
 2: <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 3:
 4:     <!-- output pretty-printed results as plain text -->
 5:     <xsl:output method="text"/>
 6:
 7:     <!-- do print titles -->
 8:     <xsl:template match="movie">
 9:
10:         <!-- output this movie's title -->
11:         <xsl:value-of select="title"/>
12:
13:         <!-- output \n -->
14:         <xsl:text>&#xA;</xsl:text>
15:
16:     </xsl:template>
17:
18:     <!-- ignore actors element and its descendants -->
19:     <xsl:template match="actors"/>
20:
21:     <!-- ignore other text in XML document (namely whitespace) -->
22:     <xsl:template match="text()"/>
23:
24: </xsl:stylesheet>
```

```
 1: <?xml version="1.0"?>
 2: <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 3:   <xsl:output method="html"/>
 4:
 5:   <!-- print each movie’s title iteratively -->
 6:   <xsl:template match="/">
 7:     <xsl:for-each select="/database/movies/movie">
 8:       <xsl:value-of select="title"/>
 9:       <br/>
10:     </xsl:for-each>
11:   </xsl:template>
12:
13: </xsl:stylesheet>
```

```
 1: <?xml version="1.0"?>
 2: <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 3:   <xsl:output method="html"/>
 4:
 5:   <!-- pass document's movie elements to printTitle -->
 6:   <xsl:template match="/">
 7:     <xsl:variable name="movies" select="/database/movies/movie"/>
 8:     <xsl:call-template name="printTitles">
 9:       <xsl:with-param name="nodes" select="$movies"/>
10:     </xsl:call-template>
11:   </xsl:template>
12:
13:   <!-- recursively print title of each movie in node-set -->
14:   <xsl:template name="printTitles">
15:     <xsl:param name="nodes"/>
16:     <xsl:value-of select="$nodes[1]/title"/>
17:     <br/>
18:     <xsl:if test="count($nodes) &gt; 1">
19:       <xsl:call-template name="printTitles">
20:         <xsl:with-param name="nodes" select="$nodes[position() &gt; 1]"/>
21:       </xsl:call-template>
22:     </xsl:if>
23:   </xsl:template>
24:
25: </xsl:stylesheet>
```

```
 1: <?xml version="1.0" encoding="iso-8859-1"?>
 2: <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 3:   <xsl:output encoding="iso-8859-1" method="text" version="1.0"/>
 4:
 5:   <!-- pass the document's children to our converter -->
 6:   <xsl:template match="/">
 7:     COUNTS
 8:
 9:       comment nodes:   <xsl:value-of select="count(//comment())"/>
10:       PI nodes:    <xsl:value-of select="count(//processing-instruction())"/>
11:       text nodes:    <xsl:value-of select="count(//text())"/>
12:       element nodes:   <xsl:value-of select="count(//*)"/>
13:       attribute nodes: <xsl:value-of select="count(//attribute::*)"/>
14:       namespace nodes: <xsl:value-of select="count(//namespace::*)"/>
15:
16:     CONTENTS
17:     <xsl:call-template name="typechecker">
18:       <xsl:with-param name="nodes" select="//child::node() | //attribute::* | //namespace::*"/>
19:     </xsl:call-template>
20:   </xsl:template>
21:
22:   <xsl:template name="typechecker">
23:     <xsl:param name="nodes"/>
24:
25:     <!-- iterate through each node in given nodeset-->
26:     <xsl:for-each select="$nodes">
27:
28:       <!-- act differently based on each node's type -->
29:       <xsl:choose>
30:
31:         <!-- if current node's an attribute, report such -->
32:         <xsl:when test="count(. | ../@*) = count(../@*)">
33:           attribute: <xsl:value-of select="name()"/>: <xsl:value-of select="."/>
34:         </xsl:when>
35:
36:         <!-- if current node's a comment, report such -->
37:         <xsl:when test="self::comment()">
38:           comment: <xsl:value-of select="."/>
39:         </xsl:when>
40:
41:         <!-- if current node's a namespace, report such -->
42:         <xsl:when test="count(. | ../namespace::*) = count(../namespace::*)">
43:           namespace: <xsl:value-of select="name()"/>: <xsl:value-of select="."/>
44:         </xsl:when>
45:
```

```
46:         <!-- if current node's a PI, report such -->
47:         <xsl:when test="self::processing-instruction()">
48:           PI: <xsl:value-of select="name()"/>: <xsl:value-of select="."/>
49:         </xsl:when>
50:
51:         <!-- if current node's text, report such -->
52:         <xsl:when test="self::text()">
53:           text: <xsl:value-of select="substring(., 0, 32)"/><xsl:if test="string-length(.) > 32">...</xsl:if>
54:         </xsl:when>
55:
56:         <!-- if current node's an element, report such -->
57:         <xsl:otherwise>
58:           element: <xsl:value-of select="name()"/>
59:         </xsl:otherwise>
60:
61:       </xsl:choose>
62:
63:     </xsl:for-each>
64:   </xsl:template>
65: </xsl:stylesheet>
```