```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="iso-8859-1" indent="yes" method="xml" version="1.0"/>

  <!-- pass the document’s children to our converter -->
  <xsl:template match="/">
    <xsl:call-template name="converter">
      <xsl:with-param name="nodes" select="child::node()"/>
    </xsl:call-template>
  </xsl:template>

  <!-- convert all attributes to child elements, leaving rest of document as-is -->
  <xsl:template name="converter">
    <xsl:param name="nodes"/>

    <!-- iterate through each node in given nodeset-->
    <xsl:for-each select="$nodes">

      <!-- act differently based on each node’s type -->
      <xsl:choose>

        <!-- else if current node’s a comment, output it as-is -->
        <xsl:when test="self::comment()">
          <xsl:comment><xsl:value-of select="."/></xsl:comment>
        </xsl:when>

        <!-- else if current node’s a PI, output it as-is -->
        <xsl:when test="self::processing-instruction()">
          <xsl:processing-instruction name="{name()}"><xsl:value-of select="."/></xsl:processing-instruction>
        </xsl:when>

        <!-- else if current node’s text, output it as-is -->
        <xsl:when test="self::text()"><xsl:value-of select="."/></xsl:when>

        <!-- else if current node’s an element, output it as-is -->
        <xsl:otherwise>
          <xsl:element name="{name()}">

            <!-- output each attribute as a child element -->
            <xsl:for-each select="attribute::*">
              <xsl:element name="{name()}"><xsl:value-of select="."/></xsl:element>
            </xsl:for-each>

            <!-- now convert this node’s children -->
            <xsl:call-template name="converter">
```

```
              <xsl:with-param name="nodes" select="child::node()"/>
            </xsl:call-template>
          </xsl:element>
        </xsl:otherwise>

      </xsl:choose>

    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="yes"/>

  <!-- start recursive-descent processing at the root -->
  <xsl:template match="/">
    <xsl:apply-templates select="child::node()"/>
  </xsl:template>

  <!-- output elements, with attributes converted to child elements -->
  <xsl:template match="*">
    <xsl:element name="{name()}">
      <!-- convert attributes to child elements -->
      <xsl:for-each select="attribute::*">
        <xsl:element name="{name()}"><xsl:value-of select="."/></xsl:element>
      </xsl:for-each>
      <!-- preserve namespace nodes -->
      <xsl:for-each select="namespace::*">
        <!-- not easy; best to wait for XSLT 2.0 -->
      </xsl:for-each>
      <xsl:apply-templates select="child::node()"/>
    </xsl:element>
  </xsl:template>

  <!-- output text verbatim -->
  <xsl:template match="text()">
    <xsl:value-of select="."/>
  </xsl:template>

  <!-- output PIs verbatim -->
  <xsl:template match="processing-instruction()">
    <xsl:processing-instruction name="{name()}"><xsl:value-of select="."/></xsl:processing-instruction>
  </xsl:template>

  <!-- output comments verbatim -->
  <xsl:template match="comment()">
    <xsl:comment><xsl:value-of select="."/></xsl:comment>
  </xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="iso-8859-1" indent="yes" method="xml" version="1.0"/>

  <!-- copy all nodes to output, but transform attributes to child elements -->
  <xsl:template match="node()">
    <xsl:copy>
      <xsl:for-each select="attribute::*">
        <xsl:element name="{name()}"><xsl:value-of select="."/></xsl:element>
      </xsl:for-each>
      <xsl:apply-templates select="child::node()"/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="iso-8859-1"?>

<xsl:stylesheet version="1.0" xmlns:xalan="http://xml.apache.org/xslt" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <!-- output pretty-printed results as XHTML 1.0 -->
    <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"/>

    <xsl:template match="/">

        <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

            <head>
                <title>1</title>
            </head>

            <body>

                <!-- iterate over movie elements, sorted by title -->
                <xsl:for-each select="database/movies/movie">
                    <xsl:sort select="title"/>

                    <!-- output tiles of R movies in red, G movies in green, and others in black -->
                    <xsl:choose>
                        <xsl:when test="rating='R'">
                            <font color="red">
                                <xsl:value-of select="title"/>
                            </font>
                        </xsl:when>
                        <xsl:when test="rating='G'">
                            <font color="green">
                                <xsl:value-of select="title"/>
                            </font>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="title"/>
                        </xsl:otherwise>
                    </xsl:choose>

                    <br/>

                </xsl:for-each>

            </body>

        </html>

    </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <!-- output pretty-printed results as plain text -->
    <xsl:output method="text"/>

    <!-- do print titles -->
    <xsl:template match="movie">

        <!-- output this movie’s title -->
        <xsl:value-of select="title"/>

        <!-- output \n -->
        <xsl:text>&#xA;</xsl:text>

    </xsl:template>

    <!-- ignore actors element and its descendants -->
    <xsl:template match="actors"/>

    <!-- ignore other text in XML document (namely whitespace) -->
    <xsl:template match="text()"/>

</xsl:stylesheet>
```

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>

  <!-- print each movie’s title iteratively -->
  <xsl:template match="/">
    <xsl:for-each select="/database/movies/movie">
      <xsl:value-of select="title"/>
      <br/>
    </xsl:for-each>
  </xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>

  <!-- pass document’s movie elements to printTitle -->
  <xsl:template match="/">
    <xsl:variable name="movies" select="/database/movies/movie"/>
    <xsl:call-template name="printTitles">
      <xsl:with-param name="nodes" select="$movies"/>
    </xsl:call-template>
  </xsl:template>

  <!-- recursively print title of each movie in node-set -->
  <xsl:template name="printTitles">
    <xsl:param name="nodes"/>
    <xsl:value-of select="$nodes[1]/title"/>
    <br/>
    <xsl:if test="count($nodes) &gt; 1">
      <xsl:call-template name="printTitles">
        <xsl:with-param name="nodes" select="$nodes[position() &gt; 1]"/>
      </xsl:call-template>
    </xsl:if>
  </xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output encoding="iso-8859-1" method="text" version="1.0"/>

  <!-- pass the document’s children to our converter -->
  <xsl:template match="/">
    COUNTS

      comment nodes:   <xsl:value-of select="count(//comment())"/>
      PI nodes:    <xsl:value-of select="count(//processing-instruction())"/>
      text nodes:    <xsl:value-of select="count(//text())"/>
      element nodes:   <xsl:value-of select="count(//*)"/>
      attribute nodes: <xsl:value-of select="count(//attribute::*)"/>
      namespace nodes: <xsl:value-of select="count(//namespace::*)"/>

    CONTENTS
    <xsl:call-template name="typechecker">
      <xsl:with-param name="nodes" select="//child::node() | //attribute::* | //namespace::*"/>
    </xsl:call-template>
  </xsl:template>

  <xsl:template name="typechecker">
    <xsl:param name="nodes"/>

    <!-- iterate through each node in given nodeset-->
    <xsl:for-each select="$nodes">

      <!-- act differently based on each node’s type -->
      <xsl:choose>

        <!-- if current node’s an attribute, report such -->
        <xsl:when test="count(. | ../@*) = count(../@*)">
          attribute: <xsl:value-of select="name()"/>: <xsl:value-of select="."/>
        </xsl:when>

        <!-- if current node’s a comment, report such -->
        <xsl:when test="self::comment()">
          comment: <xsl:value-of select="."/>
        </xsl:when>

        <!-- if current node’s a namespace, report such -->
        <xsl:when test="count(. | ../namespace::*) = count(../namespace::*)">
          namespace: <xsl:value-of select="name()"/>: <xsl:value-of select="."/>
        </xsl:when>

        <!-- if current node’s a PI, report such -->
        <xsl:when test="self::processing-instruction()">
          PI: <xsl:value-of select="name()"/>: <xsl:value-of select="."/>
        </xsl:when>

        <!-- if current node’s text, report such -->
        <xsl:when test="self::text()">
          text: <xsl:value-of select="substring(., 0, 32)"/><xsl:if test="string-length(.) > 32">...</xsl:if>
        </xsl:when>

        <!-- if current node’s an element, report such -->
        <xsl:otherwise>
          element: <xsl:value-of select="name()"/>
        </xsl:otherwise>

      </xsl:choose>

    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```