```
++++++++++++++Defining Complex Type Schema Lesson 6++++++++++



1.)     Deriving "Anonymous" Complex Types


<xs:element name="year_built">
        <xs:complexType>
            <xs:simpleContent><!--no children, just data-->
                <xs:extension base="xs:positiveInteger"><!--must be a positive integer-->
                        <!--need complex type to define attribute-->
                    <xs:attribute name="era" type="xs:string"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
</xs:element>

Example of use in XML document

<year_built era="BC">282</year_built>


2.)     Deriving **Named**" Complex Types

    Create the definition
    <!--I can reuse this definition by calling the name-->
    <xs:complexType name="yearType">
        <xs:simpleContent>
            <xs:extension base="xs:positiveInteger">
                <xs:attribute name="era" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    Reference the definition
    <xs:complexType name="historyType">
        <xs:sequence>
            <xs:element name="year_built" type="yearType"/>
            <!--"yearType" references the above definition-->
            <xs:element name="year_destroyed" type="yearType"/>
        </xs:sequence>
    </xs:complexType>

    Here is how the Derived "named" defintion looks in XML
            <year_built era="BC">282</year_built>
            <year_destroyed era="BC">226</year_destroyed>



3.)     Anonymous Complex Types with one Child

    <xs:element name="ancient_wonders">
        <xs:complexType>
```

```
            <xs:complexContent>
                <xs:restriction base="xs:anyType">
                    <xs:sequence>
                        <xs:element name="wonder" type="wonderType"/>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    Same thing, but without restriction and base which are not needed
    (even with one child, you must use <xs:sequence>)
        Preferred:
    <xs:element name="ancient_wonders">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="wonder" type="wonderType"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>


4.)     Deriving Named Complex Types **Sequence** of Children
    <xs:complexType name="wonderType">
        <xs:sequence>
            <xs:element name="name" type="nameType"/>
            <xs:element name="location" type="xs:string"/>
            <xs:element name="height" type="heightType"/>
            <xs:element name="history" type="historyType"/>
            <xs:element name="main_image" type="imageType"/>
            <xs:element name="source" type="sourceType"/>
        </xs:sequence>
    </xs:complexType>

<!--gets referenced by element "name"-->
    <xs:complexType name="nameType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="language" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

<!--gets referenced by element "height"-->
    <xs:complexType name="heightType">
        <xs:simpleContent>
            <xs:extension base="xs:nonNegativeInteger">
                <xs:attribute name="units" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>


5.)     Allowing Child Elements to Appear in **Any** Order
```

```
<xs:complexType name="historyType">
    <!--use xs:all to allow any order-->
        <xs:all>
            <xs:element name="year_built" type="yearType"/>
            <xs:element name="year_destroyed" type="yearType"/>
            <xs:element name="how_destroyed" type="destrType"/>
            <xs:element name="story" type="storyType"/>
        </xs:all>
    </xs:complexType>


6.)     Creating Child Elements as a set of **Choices**
<!--use xs:choice to allow any order-->
            <xs:complexType name="wonderType">
                <xs:sequence>
                    <xs:element name="name"
                            type="nameType"/>
                            **THIS **
                    <xs:choice>
                        <xs:element name="location"
                            type="xs:string"/>
                    <xs:sequence>
                            **OR THIS**
                        <xs:element name="city"
                            type="xs:string"/>
                        <xs:element name="country"
                            type="xs:string"/>
                    </xs:sequence>
                    </xs:choice>
                    ...
                </xs:sequence>
            </xs:complexType>

        XML Example
        <wonder>
            <name language="English"> Colossus of Rhodes</name>
            <location>Rhodes, Greece</location>
                ...
        </wonder>


        <wonder>
            <name language="English">Colossus of Rhodes</name>
            <city>Rhodes</city>
            <country>Greece</country>
            ...
        </wonder>


7.)     Defining Elements to Contain   ***Text***  Only

<xs:complexType name="yearType">
    <xs:simpleContent>
        <xs:extension base= "xs:positiveInteger">
```

```
            <xs:attribute name="era" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xs:complexType name="historyType">
        <xs:sequence>
            <xs:element name="year_built" type="yearType"/>
            <xs:element name="year_destroyed" type="yearType"/>
            <xs:element name="how_destroyed" type="destrType"/>
            <xs:element name="story" type="storyType"/>
        </xs:sequence>
</xs:complexType>

        Example XML
        <year_built era="BC">282</year_built>   ***VALID***

        <year_built era="BC">long ago</year_built>  ***INVALID***

8.)     Defining Elements to be **EMPTY**
<xs:complexType name="sourceType">
    <xs:attribute name="sectionid" type="xs:positiveInteger"/>
    <xs:attribute name="newspaperid" type="xs:positiveInteger"/>
</xs:complexType>

        Example XML
        <source sectionid="101" newspaperid="21"/>



9.)     Defining Elements to be have ***MIXED Content***
<xs:complexType name="story" mixed="true">
    <xs:sequence>
        <xs:element name="para" maxOccurs="unbounded">
            <xs:complexType/>
        </xs:element>
    </xs:sequence>
</xs:complexType>

        ***The XML Schema attribute maxOccurs="unbounded" allows for unlimited number
                of para elements within the story element.***
        Example XML
        <story>
        In 294 BC, the people of the island
        of Rhodes began building a colossal
        statue of the sun god Helios. They
        believed that it was because of his
        blessings that they were able to
        withstand a long siege on the
        island and emerge victorious.
        <para/>
        The Colossus was built with bronze,
        reinforced with iron, and weighted
        with stones. While it is often
```

```
217          depicted straddling Mandrákion
218          harbor, this is now considered
219          technically impossible; and
220          therefore, it likely stood beside
221          the harbor.
222          <para/>
223          The statue was toppled by an
224          earthquake in 226 BC. ...
225          </story>
226
227  9a.)        Another example of mixed content with an attribute that indicates the
     number of times it can occur
228  <!--mixed content named complex type-->
229      <xs:complexType name="notesType" mixed="true">
230          <xs:sequence minOccurs="1">
231              <xs:element name="major" type="xs:string">
232              </xs:element>
233          </xs:sequence>
234      </xs:complexType>
235
236
237  10.)        Deriving Complex Types from Existing Complex Types
238
239      <xs:complexType name="wonderType">
240          <xs:sequence>
241              <xs:element name="name" type="nameType"/>
242              <xs:element name="location" type="xs:string"/>
243              <xs:element name="height" type="heightType"/>
244              <xs:element name="history" type="historyType"/> <!--reference-->
245              <xs:element name="main_image" type="imageType"/>
246              <xs:element name="source" type="sourceType"/>
247          </xs:sequence>
248      </xs:complexType>
249
250      <xs:complexType name="historyType">
251          <xs:sequence>
252              <!--all of these get added as children to the element "history" above-->
253              <xs:element name="year_built" type="yearType"/>
254              <xs:element name="year_destroyed" type="yearType"/>
255              <xs:element name="how_destroyed" type="destrType"/>
256              <xs:element name="story" type="storyType"/>
257          </xs:sequence>
258      </xs:complexType>
259
260  11.)        Defining 3 Attributes, one with restrictions
261  <xs:complexType name="sourceType">
262          <xs:attribute name="sectionid" type="xs:positiveInteger"/>
263          <xs:attribute name="newspaperid">
264              <xs:simpleType><!--just data-->
265                  <xs:restriction base="xs:positiveInteger">
266                  <!--exactly 4 digit pattern using regular expressions-->
267                      <xs:pattern value="\d{4}"/>
268                  </xs:restriction>
269              </xs:simpleType>
```

```
        </xs:attribute>
    </xs:complexType>

Example of **attribute definition** in XML
    Invalid:
        <source sectionid="101" newspaperid="21"/>
    Valid:
        <source sectionid="101" newspaperid="1321"/>

12.) Defining Attributes, and making them required
        <xs:complexType name="nameType">
            <xs:simpleContent>
                <xs:extension base="xs:string">
                    <xs:attribute name="id" type="xs:integer" use="required"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
```