**Exercice 1**

```
<xs:simpleType>
    <xs:restriction base="xs:time ">
      <xs:minInclusive value="02 :30 :00"/>
      <xs:maxInclusive value="16 :50 :00"/>
    </xs:restriction>
  </xs:simpleType>
```

Commentaire :

xs:time, puisqu'il s'agit de l'heure

```
<xs:simpleType>
    <xs:restriction base="xs:double ">
      <xs:minInclusive value="-3476.4 "/>
      <xs:maxExclusive value="5"/>
    </xs:restriction>
  </xs:simpleType>
```

Commentaire :

maxExclusive car strictement inférieur à 5

```
<xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value=" [a-z] [a-z]   [a-z]
[a-z] "/>
    </xs:restriction>
  </xs:simpleType>
```

Ou

```
<xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="4"/>
    </xs:restriction>


  </xs:simpleType>
<xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value=" jpg"/>
      <xs:enumeration value="gif"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
```

Commentaire :

xs:enumeration , ressemble à ceci :

jpg| gif| BMW

```
<xs:simpleType>
    <xs:restriction base="xs:positiveinteger ">
      <xs:pattern value=" ( [a-z]{13} ){2} "/>

    </xs:restriction>
  </xs:simpleType>
```

Commentaire :

[a-z]{13}, les lettres sont repetées 13 fois

( [a-z]{13} ) {2},  les 13 lettres sont repêtées 2fois

**Exercice 2**

XML TO SCHEMA  XML :

```
<neufs>
    <item prix="18000">
        <marque>Renault</marque>
        <modele>Clio IV</modele>
    </item>
    <item prix="29900">
        <marque>BMW</marque>
        <modele>Serie 5</modele>
    </item>
</neufs>
```

Commentaires :

Chaque element contenant d'autres élements est  un complextype  et va avoir un nom

```
<xs:complexType name="voiture">
```

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

Et les elements composants l'element complexe contituent une sequence :

```
        <xs : Sequence>

              <xs:element name="marque" type="xs:string"/>
                            <xs:element name="modele"
              type="xs:string"/>



              </xs:sequence>
```

Si ces elements ont des attributs ; ils seront ajoutés comme suit :

```
              <xs:attribute name="prix"
              type="xs:positiveInteger"/>
```

si on met l'ensemble, on aura ceci :

```
< Xs :complextype
      name= ‘les_voitures’>

  <xs:sequence>
              <xs:element
  name="marque" type="xs:string"/>
              <xs:element
  name="modele" type="xs:string"/>
          </xs:sequence>
          <xs:attribute name="prix"
  type="xs:positiveInteger"/>
      </xs:complexType>
```

Commentaire :

Vu que  la balise occases contient les données de complextype les_voitures  , de ce fait on fera l’heritage de types comme suit avec le mot clé :

```
<xs:extension base=" les_voitures">



</xs : extension>
```

En gros on aura ceci :

```
                < Xs :complextype
                     name= ‘les_voitures’>

                  <xs:sequence>
                              <xs:element
                  name="marque" type="xs:string"/>
                              <xs:element
                  name="modele" type="xs:string"/>
                          </xs:sequence>
                          <xs:attribute name="prix"
                  type="xs:positiveInteger"/>
                      </xs:complexType>


<xs:complexType name="occases">


<xs:complexContent>

<xs:extension base=" les_voitures">

   <xs :sequence>



   <xs:element name="annee"
   type="xs:positiveInteger"/>







   </xs:sequence>
```

```
</xs : extension>

</xs:complexContent>

</xs:complexType>
```

Commentaire :

Vu qu'il y  a la balise année, de ce  fait, on ajoute

```
<xs:element name="annee"
type="xs:positiveInteger"/>
```

Pour avoir ceci :

```
<xs:complexType name="occases">
        <xs:complexContent>
            <xs:extension
  base="voiture">
                <xs:sequence>
                    <xs:element
  name="annee"
  type="xs:positiveInteger"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

Metenant occupont nous de  balises proprement dites qui seront des elements  de notre  xsd :

```
<xs:element name="occases">


</xs:element>
<xs:element name="neufs ">


</xs:element>


<xs:element name="stock ">


</xs:element>
```

Commentaire :

Les elements de xsd sont que les  grandes balises

Occupond nous  de leurs contenus rien de surprenant on a

```
<xs:complexType>


</xs:complexType>
```

Car ces balises contiennent d’autres balises

Et que les balises occases et neufs contiennent une suite d'élements les items de ce fait ces items forment une sequence :

```
<xs:sequence>
<xs :element name='item'  type='occases'
minOccurs="0" maxOccurs="unbounded"
</xs:sequence>
```

Commentaire :

Le type occases pour le type crée occases ci haut

Pour l'element occases, on aura :

```
<xs:element name=' occases'>

    <xs:complexType>
        <xs:sequence>
            <xs:element name="item"
type="occases"  minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Commentaire :

minOccurs="0" maxOccurs="unbounded" permet de dire que le nombre  des items est illimité  et le type occases fait reference au complextype occases

```
    <xs:element name=’neufs ‘>


        <xs:complexType>
            <xs:sequence>
                <xs:element name="item"
type=" les_voitures "  minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
```

```
<xs:element name=’stock‘>


    <xs:complexType>
        <xs:sequence>

                    <xs:element
  ref="neufs"/>

  <xs:element ref="occases"/>

        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Commentaire :

La balise stock contient les elements occases et neufs d’où <xs:element ref="occases"/>
Et <xs:element ref="neufs "/>

Vu qu’il y a un attribut nom dans la balise stock, on ajoute ceci :

```
<xs:attribute name="nom"  type="xs:string"/>
```

On aura ceci :

```
<xs:element name=’stock‘>
```

```
                <xs:complexType>
                    <xs:sequence>

                                <xs:element
            ref="neufs"/>

            <xs:element ref="occases"/>

                    </xs:sequence>


        <xs:attribute name="nom"  type="xs:string"/>

                </xs:complexType>
```

Et dans l’ensemble on aura ceci :

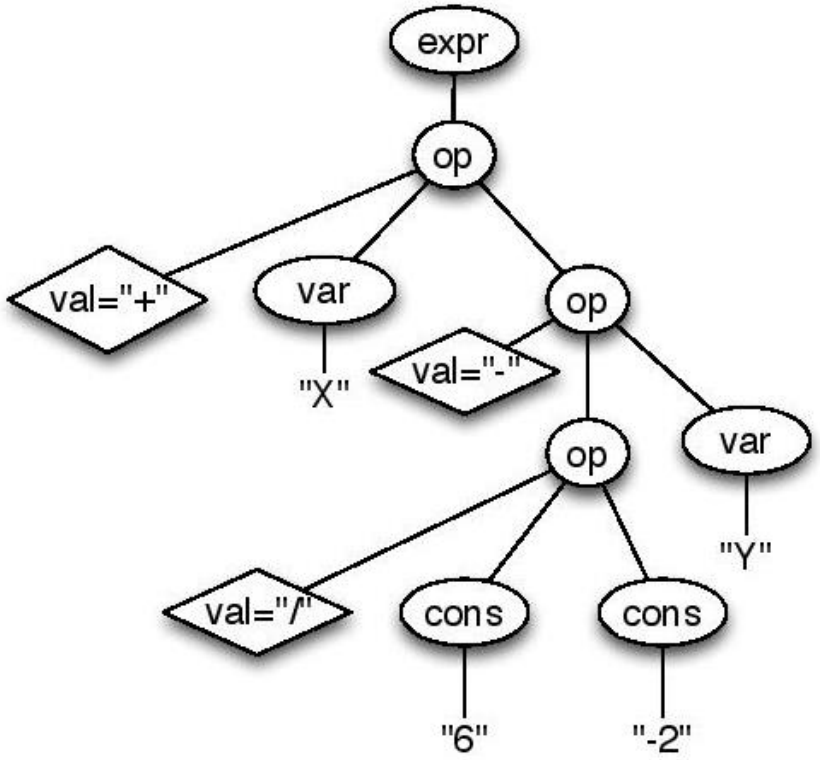
```
<?xml version="1.0" encoding="UTF-8" ?>

  <xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">


            < Xs :complextype
                 name= ‘les_voitures’>

              <xs:sequence>
                          <xs:element
              name="marque" type="xs:string"/>
                          <xs:element
              name="modele" type="xs:string"/>
                      </xs:sequence>
                      <xs:attribute name="prix"
              type="xs:positiveInteger"/>
                  </xs:complexType>
```

```
<xs:complexType name="occases">
                        <xs:complexContent>
                            <xs:extension
                base="voiture">
                                    <xs:sequence>
                                        <xs:element
                name="annee"
                type="xs:positiveInteger"/>
                                    </xs:sequence>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>


            <xs:element name=’neufs ‘>


                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="item"
        type=" les_voitures "  minOccurs="0"
        maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name=’stock‘>


                <xs:complexType>
```

```
                <xs:sequence>

                                <xs:element
            ref="neufs"/>

            <xs:element ref="occases"/>

                </xs:sequence>


      <xs:attribute name="nom"  type="xs:string"/>

            </xs:complexType>

</xsd:schema>
```

## Exercice 3

Opêration arithmetique to schema xml :




*Commentaire :*

Les operations sont : +,-,* et /

On a le choix à une operation en xsd on aura :

```
<xs:simpleType name=’operations’>


        <xs:restriction base="xs:string">
            <xs:enumeration value="+"/>
            <xs:enumeration value="-"/>
            <xs:enumeration value="/"/>
            <xs:enumeration value="*"/>
        </xs:restriction>
    </xs:simpleType>
```

Les operations portent sur les constantes ou variables :

```
<xs:complexType name=’operande’>

    <xs:choice minOccurs="2" maxOccurs="2">
                <xs:element name="cons"
    type="xs:integer"/>
                <xs:element name="var"
    type="xs:string"/>
                <xs:element name="op"
    type="Operande"/>
            </xs:choice>
            <xs:attribute name="val"
    type="typeOp" use="required"/>
        </xs:complexType>
```

Commentaire :

Vu l’arbre, les operandes peuvent être une constante, une variable ou un operateur

Chaque operande a  un attribut val

```
<xs:element
name="expr">

    <xs:complexType>
```

```
    <xs:sequence>

    <xs:element
name="op"
type="Operande"/>

    </xs:sequence>

    </xs:complexType>
    </xs:element>
```

Commentaire :

Et la racine expr qui est une suite d’operandes

Exercice 4 :

```
<xs:complexType  name=’ types_clb ’>

          <xs:sequence>
              <xs:element name="nom"
  type="xs:string"/>
              <xs:element name="ville"
  type="xs:string"/>
          </xs:sequence>
          <xs:attribute name="id" type="xs:string"
  use="required"/>
      </xs:complexType>
```

**Commentaire :**

Chaque club est une suite de nom et ville d'où le complextype dedans les elements villes et noms et l'attribut id de club

```
<xs:element name='clubs'> <xs:complexType>
                    <xs:sequence>
                        <xs:element
                name="club" type=" types_clb "
                minOccurs="20" maxOccurs="20"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
```

**Commentaire :**

La balise clubs est une suite de clubs d'où

```
<xs:sequence>
    <xs:element name="club" type="
types_clb " minOccurs="20" maxOccurs="20"/>
</xs:sequence>
```

maxOccurs="20", car il y a 20 clubs

```
        <xs:complexType
        name=’type_rencontre’>


<xs:sequence>
            <xs:element name="clubReceveur"
type="xs:string"/>
            <xs:element name="clubInvité"
type="xs:string"/>
            <xs:element name="score"
type="xs:string"/>
        </xs:sequence>
    </xs:complexType>


 <xs:complexType
 name=’journees’>

                <xs:sequence>

                        <xs:element
                name="date"
                type="xs:date"/>
                        <xs:element
                name="rencontre"
                type="typeRencontre"
                minOccurs="0"
                maxOccurs="10"/>
                    </xs:sequence>
                    <xs:attribute
                name="num"
                type="xs:integer"
                use="required"/>
```

```
                    </xs:complexType>

<xs:element
name="journées">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element
                    name="journée" type="typeJournée"
                    minOccurs="0" maxOccurs="38"/>
                            </xs:sequence>
                        </xs:complexType>
```

```
                                                    </xs:elemen
                                                    t>


<xs:element
name="championnat">
                        <xs:complexType>
                            <xs:sequence>

                            <xs:element
                        ref="clubs"/>

                            <xs:element
                        ref="journées"/>

                            </xs:sequence>

                            </xs:complexType
                            </xs:element>
```