TD/TP n° 1

Exercice 1.

On considère la DTD suivante :

```
<!DOCTYPE A [
 <!ELEMENT A (B+, C)>
 <!ELEMENT B (#PCDATA)>
 <!ELEMENT C (B?, D)>
 <!ELEMENT D (#PCDATA)>
]>
```

Pour chaque séquence d'ouverture et fermetures de balises, ci-dessous :

- Construire l'arbre XML correspondant.
- Dire si elle correspond à la DTD ci-dessus. Justifier.

1. `A C B /B B /B D /D /C B /B /A`
2. `A B /B B /B C B /B D /D /A`
3. `A B /B B /B C B /B /C /A`
4. `A B /B C B /B D /D /C /A`

Exercice 2.

On considère la portion de document XML suivante :

```
<EMP name = "Kermit">
   <ADDR>123 Sesame St.</ADDR>
   <PHONE type = "cell">555-1212</PHONE>
</EMP>
```

Pour chaque propositions d'attributs qui suivent, déterminer si elle peut faire partie de la DTD du document :

1. `<!ATTLIST PHONE type CDATA #REQUIRED>`
2. `<!ATTLIST EMP ssNo CDATA #REQUIRED>`
3. `<!ATTLIST ADDR zip CDATA #IMPLIED>`
4. `<!ATTLIST EMP ssNo ID #IMPLIED>`

Exercice 3.

On considère la portion de document XML suivante :

```
<INFO>
   <ADDR>101 Maple St.</ADDR>
   <PHONE>555-1212</PHONE>
   <PHONE>555-4567</PHONE>
</INFO>
```

Laquelle des propositions suivantes peut-être la spécification de l'élément INFO dans la DTD que ce document respecte ?

1. `<!ELEMENT INFO (ADDR,NAME+,PHONE*)>`
2. `<!ELEMENT INFO (ADDR,PHONE)>`
3. `<!ELEMENT INFO (ADDR*,PHONE+,MANAGER)>`
4. `<!ELEMENT INFO (ADDR*,PHONE+)>`

Exercice 4. On considère la DTD suivante :

```
<!DOCTYPE meal [
   <!ELEMENT meal (person*,food*,eats*)>
   <!ELEMENT person EMPTY>
   <!ELEMENT food EMPTY>
   <!ELEMENT eats EMPTY>
   <!ATTLIST person name ID #REQUIRED>
   <!ATTLIST food name ID #REQUIRED>
   <!ATTLIST eats diner IDREF #REQUIRED
                  dish IDREF #REQUIRED>
]>
```

Les documents suivants correspondent-ils à la DTD ?

1.
```
<meal>
   <person name="Alice"/>
   <food name="salad"/>
   <eats diner="Alice" dish="salad"/>
   <person name="Bob"/>
   <food name="salad"/>
   <eats diner="Bob" dish="salad"/>
   <person name="Carol"/>
   <food name="sandwich"/>
   <eats diner="Carol" dish="sandwich"/>
</meal>
```
2.
```
<meal>
  <person name="Alice"/>
  <person name="Bob"/>
  <food name="salad"/>
  <eats diner="Alice" dish="food"/>
  <eats diner="Bob" dish="food"/>
</meal>
```
3.
```
<meal>
  <person name="Alice"/>
  <person name="Bob"/>
  <person name="Carol"/>
  <person name="Dave"/>
  <food name="salad"/>
  <food name="turkey"/>
  <food name="sandwich"/>
  <eats diner="Alice" dish="turkey"/>
  <eats diner="Bob" dish="salad"/>
  <eats diner="turkey" dish="Dave"/>
</meal>
```

Exercice 5.

On s'intéresse au document XML bien formé qui satisfait les conditions :
- Il a un élément racine « tasklist ».
- L'élément racine a 3 « task » comme sous-éléments.
- Chacun des sous-éléments « task » a un attribut nommé « name ».
- Les valeurs de l'attribut « name » sont « eat », « drink » et « play ».

1. Sélectionnez, à partir des choix suivants, le document XML bien formé qui satisfait les conditions précédentes :

```
<tasklist>
  <task name="eat"/>
</tasklist>
<tasklist>
   <task name="drink"/>
</tasklist>
<tasklist>
  <task name="play"/>
</tasklist>
```

```
<tasklist>
    <task name="eat"></task>
    <task name="drink"></task>
    <task name="play"></task>
 </tasklist>
```

```
<tasklist>
    <task name=eat/>
    <task name=drink/>
    <task name=play/>
  </tasklist>
```

```
<tasklist>
     <task name="eat"/>
     <task name="drink"/>
     <task name="play"/>
  <tasklist>
```

2. Construire une DTD qui assure les conditions décrites au dessus.

Exercice 6. TP

On travaillera sous Linux. On a besoin d'un terminal et d'un bon éditeur de texte.
Si vous n'en avez pas encore de favori, je vous conseille `kate`*, qui dispose de la coloration syntaxique, d'un terminal intégré et qui peut se scinder ; pratique pour avoir d'un côté le XML et de l'autre la DTD !*

1. On veut construire une DTD correspondant à un document XML fourni, `courses-noID.xml`.
   a) Récupérer le document à l'aide de la commande :
   ```
   wget http://www.lacl.fr/~msassolas/enseignement/XML_Licence/courses-noID.xml
   ```
   b) Construire une DTD `courses-noID.dtd` pour ce document.
   c) On vérifiera que la DTD correspond à l'aide de la commande :
   ```
   xmllint --dtdvalid courses-noID.dtd courses-noID.xml
   ```
   Elle doit renvoyer le XML validé (et les erreurs, le cas échéant). La commande :
   ```
   xmllint --noout --dtdvalid courses-noID.dtd courses-noID.xml
   ```
   ne renvoie rien pourvu que le XML soit valide.
2. On veut construire une DTD correspondant à un document XML fourni, `courses-ID.xml`.
   a) Récupérer le document à l'aide de la commande :
   ```
   wget http://www.lacl.fr/~msassolas/enseignement/XML_Licence/courses-ID.xml
   ```
   b) Construire une DTD `courses-ID.dtd` pour ce document.
   c) On vérifiera que la DTD correspond à l'aide de la commande :
   ```
   xmllint --dtdvalid courses-ID.dtd courses-ID.xml
   ```
   Elle doit renvoyer le XML validé (et les erreurs, le cas échéant). La commande :
   ```
   xmllint --noout --dtdvalid courses-ID.dtd courses-ID.xml
   ```
   ne renvoie rien pourvu que le XML soit valide.
3. On veut construire une DTD correspondant à un document XML fourni, `countries.xml`.
   a) Récupérer le document à l'aide de la commande :
   ```
   wget http://www.lacl.fr/~msassolas/enseignement/XML_Licence/countries.xml
   ```
   b) Construire une DTD `countries.dtd` pour ce document.
   c) On vérifiera que la DTD correspond à l'aide de la commande :
   ```
   xmllint --dtdvalid countries.dtd countries.xml
   ```
   Elle doit renvoyer le XML validé (et les erreurs, le cas échéant). La commande :
   ```
   xmllint --noout --dtdvalid countries.dtd countries.xml
   ```
   ne renvoie rien pourvu que le XML soit valide.