Examen intra – INF4375 – Automne 2009
30% de la note finale
21 octobre 2009 18h00 à 21h00

Aucune documentation n'est permise. Répondez dans le cahier fourni à cet effet.

Question 1 – 2 pts

Décrivez les différentes étapes effectuées par le « stub » et le « skeleton » lors d'un appel RPC. Les étapes doivent être décrites dans le bon ordre.

Question 2 – 2 pts

Indiquez si le document XML suivant est mal-formé et pourquoi.

```
<?xml version="1.0" encoding="utf-8"?>
<house><kitchen><refrigerator>Daewoo</refrigerator><oven>KitchenAid</oven>
</kitchen><laundry><washing-machine>Whirlpool
</washing-machine><dryer>Whirlpool</dryer><room><air-conditioner>LG
</air-conditioner></room><living-room><tv>Samsung</tv>
</living-room></laundry></house>
```

Question 3 – 4 pts

À partir du document XML suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns:jb="http://www.jberger.org/">
      <ria>Silverlight</ria>
      <jb:xhtml xmlns:s3="ca.uqam.inf4375.09.03">
            <s3:test />
            <ant xmlns="http://www.jberger.org/ant/v0.01">
                  <xml-tech>Ant</xml-tech>
            </ant>
            <jb:new>...</jb:new>
            <again>...</again>
      </jb:xhtml>
</feed>
```

Indiquez le namespace complet (et non pas le préfix) des éléments suivants :

1. feed
2. ria
3. xhtml
4. test
5. ant
6. xml-tech
7. new
8. again

Question 4 – 5 pts

À partir du document XML suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<grocery>
      <categorie id="1" name="Pâtes" />
      <categorie id="2" name="Asiatique" />
      <categorie id="3" name="Boucherie" />
      <categorie id="4" name="Charcuterie" />
      <categorie id="5" name="Bière" />
      <product categorie="5">
            <name>Heineken - 12 cannettes</name>
            <price>18.99</price>
            <sku>9362781095732</sku>
      </product>
      <product categorie="1">
            <name>Catelli Spaghettini</name>
            <price>2.45</price>
            <sku>0094326772419</sku>
      </product>
      <product categorie="3">
            <name>Filet mignon - 50 lbs</name>
            <price>245.99</price>
            <sku>7775439875421</sku>
      </product>
      <product categorie="2">
            <name>Vermicelles de riz</name>
            <price>3.99</price>
            <sku>1113828750095</sku>
      </product>
</grocery>
```

Évaluez les expressions XPath suivantes :

1. `/grocery/categorie[not(//product/@categorie = @id)]/@name`
2. `//product[price > 100.00]/following-sibling::product/name`
3. `/grocery/categorie[position() = 3]/@name`
4. `//*[price = '2.45']/../*[position() = 6]/name`
5. `/*/*[position() = last()]/*[position() = last()]`

Question 5 – 2 pts

Vous devez intégrer la possibilité de lire et traiter un document XML dans un logiciel déjà existant. Le logiciel en question souffre de problèmes de performance et il est important que la vitesse d'exécution soit optimale pour ne pas alourdir la situation. Par ailleurs, le document XML peut contenir un élément « repository » sous l'élément racine. Si tel est le cas, il est inutile de continuer la lecture du document, il est donc possible d'optimiser les performances avec ce cas très précis. De plus, la taille du document XML en question peut varier de 50 Mo à 750 Mo.

En considérant ces exigences, quel parser XML entre DOM, SAX et StAX allez-vous choisir et pourquoi?

Question 6 – 5 pts

Appliquez la feuille de style suivante sur le document XML suivant. Indiquez le résultat produit.

Feuille de style :

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="text" />
      <xsl:template match="/">
            <xsl:apply-templates select="*" />
      </xsl:template>
      <xsl:template match="entities">
            <xsl:apply-templates select="table" mode="creation">
                  <xsl:sort select="@name" />
            </xsl:apply-templates>
      </xsl:template>
      <xsl:template match="table" mode="creation">
            <xsl:text>CREATE TABLE </xsl:text>
            <xsl:value-of select="@name" />
            <xsl:text> (
ID_</xsl:text>
            <xsl:value-of select="@name" />
            <xsl:text> NUMBER(38,0),
</xsl:text>
            <xsl:for-each select="column">
                  <xsl:sort select="@name" />
                  <xsl:value-of select="@name" />
                  <xsl:text> VARCHAR2(50)</xsl:text>
                  <xsl:if test="position() != last()">
                        <xsl:text>,</xsl:text>
                  </xsl:if>
                  <xsl:text>
</xsl:text>
            </xsl:for-each>
            <xsl:text>)
</xsl:text>
      </xsl:template>
</xsl:stylesheet>
```

Document XML :

```
<?xml version="1.0" encoding="utf-8"?>
<entities>
      <table name="employee">
            <column name="firstname" />
            <column name="lastname" />
            <column name="phone" />
      </table>
      <table name="timesheet">
            <column name="year" />
            <column name="week" />
            <column name="employee_id" />
      </table>
</entities>
```

Question 7 – 2 pts

La grande majorité des outils de génération de schémas XSD utilisent la même forme de design pour les schémas générés. Quelle est cette forme de design et pourquoi son utilisation est-elle récurrente avec les générateurs de code?

Question 8 – 3 pts

Indiquez trois choses qui sont possibles de faire avec XSD et RELAX NG mais qui ne sont pas possibles avec DTD.

Question 9 – 2 pts

Expliquez en détail deux techniques servant à améliorer la fiabilité d'un service.

Question 10 – 3 pts

Définissez ce qu'est un « mashup ». De plus, si un service est un « mashup », expliquez le modèle correspondant dans l'architecture orientée-service. Illustrez le modèle à l'aide d'un schéma.