# Ejercicios de XSLT

## Ejercicio 1

Para los ejercicios 1 al 7, utilizaremos este documento XML de ejemplo y su DTD. Recuerda que debes modificar en el documento XML la URL de la segunda línea, para que apunte al lugar donde se encuentra el DTD.

Hacer una hoja de estilo que copie el contenido de cualquier documento XML a uno nuevo, utilizando `xsl:copy-of`.

## Ejercicio 2

Hacer una hoja de estilo que copie el contenido de cualquier documento XML a uno nuevo, utilizando `xsl:copy`.

## Ejercicio 3

Hacer una hoja de estilo que cree un documento XML que se componga de dos copias del documento fuente, la primera rodeada de la etiqueta `uno` y la segunda de la etiqueta `dos`. El elemento raiz del documento destino se llamará `raiz`.

## Ejercicio 4

Modifique la hoja de estilo del ejercicio 2 para que sólo se copien los elementos que sean el segundo hijo elemento de algún elemento. No se copia el contenido de los elementos copiados.

## Ejercicio 5

Modifique la hoja de estilo del ejercicio 2 para que sólo se copien los elementos que sean hijo único (excluidos nodos de texto que contengan exclusivamente espacios en blanco, saltos de línea, etc.) de algún elemento. No se copia el contenido de los elementos copiados.

## Ejercicio 6

Hacer una hoja de estilo que dado un documento fuente, obtenga un documento destino compuesto por los mismos elementos que el documento fuente. En cada elemento del documento destino habrá un atributo, llamado `descendientes`, cuyo valor es el número de elementos que contiene el elemento.

Para este ejercicio le pueden resultar útiles las siguientes funciones XSL:

- `format-number(....,'#00.0')`: permite dar formato a un número antes de ser escrito en el documento destino. El primer argumento es el número al que se le va a dar formato. El segundo argumento especifica el formato. Los caracteres `#` indican un dígito, pero si no es necesario no se escribirá nada. Los carácteres `0` indican un dígito, pero en el caso de que el dígito no sea necesario, se escribirá el dígito `0`. El `.` separa pa parte entera de la parte decimal. Por ejemplo, con el formato indicado, el `7` se representará como `07.0`.

- `count(node-set)`: indica el número de nodos que contiene un node-set.

# Ejercicio 7

Hacer una hoja de estilo que cree un documento HTML que contenga una lista. Esta lista sigue la estructura del documento XML, de manera que el texto de un item es el nombre de un elemento del documento fuente, y los subitems se corresponden con el contenido de dicho elemento.

Para el documento XML que se proporciona en el ejercicio 1, el resultado debería ser el siguiente:

```
<html>
<head>
<title> Estructura de un documento XML </title>
</head>
<body>
<ul>
<li>R</li>
<ul>
  <li>D</li>
  <li>E</li>
<ul>
    <li>E</li>
<ul>
      <li>A</li>
      <li>A</li>
    </ul>
    <li>E</li>
<ul>
      <li>A</li>
      <li>A</li>
    </ul>
  </ul>
</ul>
</ul>
</body>
</html>
```

Puede utilizar la función XPath `local-name(node-set?)` que devuelve la parte local del nombre del primer elemento del node-set que se pasa como argumento o la del current nodo si no se pasa ningún argumento.

# Ejercicio 8

En este ejercicio vamos a convertir un documento que tiene una estructura tipo árbol en una estructura tipo lista. El DTD del documento de partida lo puede encontrara aquí y el del documento de destino aquí. Aquí hay un documento de ejemplo y aquí el resultado que nos gustaría obtener.

# Ejercicio 9

En este ejercicio trabajaremos con el siguiente DTD para representar el resultado de unas elecciones, almacenado en el fichero `elecciones.dtd`:

```
<!ELEMENT Elecciones (Partido)+>
<!ATTLIST Elecciones
tipo CDATA #REQUIRED
año CDATA #REQUIRED>
```

```
<!ELEMENT Partido (Provincia)+>
<!ATTLIST Partido
nombre CDATA #REQUIRED
siglas CDATA #REQUIRED>

<!ELEMENT Provincia (#PCDATA)>
<!ATTLIST Provincia
nombre CDATA #REQUIRED>
```

El siguiente fichero XML es un ejemplo que utiliza este DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Elecciones SYSTEM "elecciones.dtd">

<Elecciones tipo="europeas" año="2014">
  <Partido nombre="Partido Popular" siglas="PP">
    <Provincia nombre="A Coruña">129814</Provincia>
    <Provincia nombre="Álava">16978</Provincia>
    <Provincia nombre="Albacete">50549</Provincia>
  </Partido>
  <Partido nombre="Partido Socialista Obrero Español" siglas="PSOE">
    <Provincia nombre="A Coruña">85117</Provincia>
    <Provincia nombre="Álava">16115</Provincia>
    <Provincia nombre="Albacete">41017</Provincia>
  </Partido>
</Elecciones>
```

Apartado 1

Escriba una hoja de estilo XSLT que dado un documento XML válido según el DTD indicado al principio de este ejercicio produzca como resultado un documento XML en el que los resultados electorales estén organizados por provincias, y dentro de éstas, se desglosen los resultados obtenidos por cada partido.

Se puede suponer que todos los elementos `Partido` contienen datos para las mismas provincias.

Por ejemplo, para el documento XML que se da al principio de este ejercicio, el resultado que se desea obtener es el siguiente:

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

<Elecciones tipo="europeas" año="2014">
  <Provincia nombre="A Coruña">
    <Partido nombre="Partido Popular"
             siglas="PP">129814</Partido>
    <Partido nombre="Partido Socialista Obrero Español"
             siglas="PSOE">85117</Partido>
  </Provincia>
  <Provincia nombre="Álava">
    <Partido nombre="Partido Popular"
             siglas="PP">16978</Partido>
    <Partido nombre="Partido Socialista Obrero Español"
             siglas="PSOE">16115</Partido>
  </Provincia>
  <Provincia nombre="Albacete">
    <Partido nombre="Partido Popular"
             siglas="PP">50549</Partido>
    <Partido nombre="Partido Socialista Obrero Español"
             siglas="PSOE">41017</Partido>
  </Provincia>
</Elecciones>
```

Apartado 2

Escriba una hoja de estilo XSLT que dado un documento XML válido según el DTD indicado al principio de este ejercicio produzca como resultado un documento XML en el que se muestren los votos totales obtenidos por cada partido.

Por ejemplo, para el documento XML que se da al principio de este ejercicio, el resultado que se desea obtener es el siguiente:

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

<Elecciones tipo="europeas" año="2014">
  <Partido nombre="Partido Popular"
           siglas="PP">197341</Partido>
  <Partido nombre="Partido Socialista Obrero Español"
           siglas="PSOE">142249</Partido>
</Elecciones>
```

**Nota 1**: En los dos apartados se deben preservar los atributos originales de los elementos que se insertan en el documento destino.

**Nota 2**: para la realización de los dos apartados que componen este ejercicio se puede partir del siguiente esqueleto:

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

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

<xsl:output method="xml" encoding="UTF-8"
  indent="yes"/>

<xsl:strip-space elements="*" />

...

</xsl:stylesheet>
```

# Ejercicio 10

Ahora vamos a trabajar con un DTD que permite definir expresiones aritméticas. El DTD lo podemos encontrar aquí. Puede probar usando este ejemplo.

Hacer una hoja de estilo que dado un documento XML que utilice nuestro DTD de expresiones aritméticas obtenga un documento XML con el mismo DTD en el que las expresiones estén evaluadas.

# Ejercicio 11

Una empresa que se dedica a la venta de productos de electrónica de consumo almacena los datos de las existencias así como de los pedidos recibidos en XML. Para ello se utiliza el siguiente DTD, que se almacena en un fichero llamado `pedidos.dtd`:

```
<!ELEMENT Facturacion (Stock, (Pedidos|Facturado)? )>

<!ELEMENT Facturado EMPTY>
```

```
<!ATTLIST Facturado
totalFacturado CDATA #REQUIRED>

<!ELEMENT Stock (Producto)+>

<!ELEMENT Producto (NombreProd, DescripcionProd?)>
<!ATTLIST Producto
referencia ID #REQUIRED
precio CDATA #REQUIRED
unidades CDATA #REQUIRED>

<!ELEMENT NombreProd (#PCDATA)>

<!ELEMENT DescripcionProd ANY>

<!ELEMENT Pedidos (Pedido)*>

<!ELEMENT Pedido (Cliente, Item+) >

<!ELEMENT Cliente  (Nombre, Apellidos, Direccion, Telefono?)>
<!ATTLIST Cliente
nif ID #REQUIRED>

<!ELEMENT Nombre (#PCDATA)>

<!ELEMENT Apellidos (#PCDATA)>

<!ELEMENT Direccion (#PCDATA)>

<!ELEMENT Telefono (#PCDATA)>

<!ELEMENT Item EMPTY>
<!ATTLIST Item
referencia IDREF #REQUIRED
cantidad CDATA "1">

<!ELEMENT b (#PCDATA)>
```

El elemento `Facturacion` actúa como elemento raíz de los documentos XML utilizados y contiene 2 elementos. El primero es un elemento `Stock` con los datos de las existencias iniciales de cada producto. El segundo elemento puede ser o bien un elemento `Pedidos` con los pedidos que se han recibido en determinado periodo de tiempo o bien un elemento `Facturado` que contiene un atributo `totalFacturado` que indica la cantidad de dinero facturada por los pedidos recibidos en cierto periodo de tiempo. Cuando el documento XML contiene un elemento `Pedidos`, se entiende que los datos de existencias indicados en el elemento `Stock` son anteriores a procesar los pedidos indicados en el elemento `Pedidos`.

Para cada `Producto` se indica una referencia interna utilizada en la empresa (atributo `referencia`), su precio (atributo `precio`) y el número de unidades de que se dispone (atributo `unidades`). Asimismo, se indica el nombre del producto (elemento `NombreProd`) y, opcionalmente, una descripción más detallada del mismo (elemento `DescripcionProd`).

Para cada `Pedido` se proporcionan los datos del `Cliente` que realiza el pedido (nombre, apellidos, nif, dirección, teléfono), así como la lista de productos (elementos `Item`) que ha adquirido el cliente. Para cada `Item` se indica la referencia interna utilizada en la empresa para el producto (atributo `referencia`) y el número de unidades adquiridas (atributo `cantidad`).

A continuación se muestra un ejemplo de un documento XML **válido** de acuerdo con este DTD.

```
<?xml version="1.0"?>
```

```
<!DOCTYPE Facturacion SYSTEM "pedidos.dtd">

<Facturacion>
  <Stock>
    <Producto referencia="r1234"
      precio="12" unidades="100">
      <NombreProd>Memoria microSD 8 GB</NombreProd>
    </Producto>
    <Producto referencia="r1235"
      precio="35" unidades="100">
      <NombreProd>Memoria microSD 16 GB</NombreProd>
    </Producto>
    <Producto referencia="r1233"
      precio="5" unidades="200">
      <NombreProd>Memoria microSD 4 GB</NombreProd>
    </Producto>
    <Producto referencia="r2101"
      precio="700" unidades="10">
      <NombreProd>Ordenador tipo netbook</NombreProd>
      <DescripcionProd>
       Intel Atom, 1600MHz, <b>2GB de memoria RAM</b>,
       disco duro 320 GB, 2 puertos USB, Fast Ethernet.
      </DescripcionProd>
    </Producto>
  </Stock>
  <Pedidos>
    <Pedido>
      <Cliente nif="n23852788F">
       <Nombre>Francisco</Nombre>
       <Apellidos>Ochoa Prieto</Apellidos>
       <Direccion>C/Pez, 7, Fuenlabrada</Direccion>
       <Telefono>612345673</Telefono>
      </Cliente>
      <Item referencia="r2101" cantidad="1" />
      <Item referencia="r1235" cantidad="2" />
      <Item referencia="r1233" cantidad="5" />
    </Pedido>
    <Pedido>
      <Cliente nif="n16755322G">
        <Nombre>Vanessa</Nombre>
        <Apellidos>Nieto Escudero</Apellidos>
        <Direccion>C/Flor, 21, Madrid</Direccion>
      </Cliente>
      <Item referencia="r1235" cantidad="1" />
      <Item referencia="r1234" cantidad="10" />
    </Pedido>
  </Pedidos>
</Facturacion>
```

**Apartado 1**

Escriba una hoja de estilo XSLT que, dado un documento XML que se ajuste al DTD presentado en el ejercicio y que contiene un elemento `Pedidos`, produzca como resultado un documento XML en el que se han realizado **exclusivamente** los siguientes cambios respecto del documento XML de entrada:

- Se actualizan los datos de existencias. Es decir, se actualiza el valor de cada atributo `unidades` de los elementos `Producto` restándole el número de unidades de dicho producto que hayan sido adquiridas en los pedidos contenidos por el elemento `Pedidos` del documento fuente.
- El elemento `Pedidos` pasa a estar vacío.

El resultado de procesar el documento XML presentado en este ejercicio con dicha hoja de estilo

XSLT se presenta a continuación.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Facturacion SYSTEM "pedidos.dtd">
<Facturacion>
  <Stock>
    <Producto unidades="90" referencia="r1234" precio="12">
      <NombreProd>Memoria microSD 8 GB</NombreProd>
    </Producto>
    <Producto unidades="97" referencia="r1235" precio="35">
      <NombreProd>Memoria microSD 16 GB</NombreProd>
    </Producto>
    <Producto unidades="195" referencia="r1233" precio="5">
      <NombreProd>Memoria microSD 4 GB</NombreProd>
    </Producto>
    <Producto unidades="9" referencia="r2101" precio="700">
      <NombreProd>Ordenador tipo netbook</NombreProd>
      <DescripcionProd>
       Intel Atom, 1600MHz, <b>2GB de memoria RAM</b>,
       disco duro 320 GB, 2 puertos USB, Fast Ethernet.
      </DescripcionProd>
    </Producto>
  </Stock>
  <Pedidos/>
</Facturacion>
```

**Apartado 2**

Escriba una hoja de estilo XSLT que, dado un documento XML que se ajuste al DTD presentado en el ejercicio y que contiene un elemento `Pedidos`, produzca como resultado un documento XML en el que se han realizado **exclusivamente** los siguientes cambios respecto del documento XML de entrada:

- Se actualizan los datos de existencias. Es decir, se actualiza el valor de cada atributo `unidades` de los elementos `Producto` restándole el número de unidades de dicho producto que hayan sido adquiridas en los pedidos contenidos por el elemento `Pedidos` del documento fuente.

- No hay elemento `Pedidos`. En su lugar, se introduce un elemento `Facturado` en el que se guarda en el atributo `totalFacturado` el importe total de los pedidos incluidos en el elemento `Pedidos` del documento fuente.

Además, la hoja de estilo XSLT deberá cumplir que cada nodo `Producto` del documento fuente se procese una sola vez.

El resultado de procesar el documento XML presentado en este ejercicio con dicha hoja de estilo XSLT se presenta a continuación.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Facturacion SYSTEM "pedidos.dtd">
<Facturacion>
  <Stock>
    <Producto unidades="90" referencia="r1234" precio="12">
      <NombreProd>Memoria microSD 8 GB</NombreProd>
    </Producto>
    <Producto unidades="97" referencia="r1235" precio="35">
      <NombreProd>Memoria microSD 16 GB</NombreProd>
    </Producto>
    <Producto unidades="195" referencia="r1233" precio="5">
      <NombreProd>Memoria microSD 4 GB</NombreProd>
    </Producto>
    <Producto unidades="9" referencia="r2101" precio="700">
```

```
      <NombreProd>Ordenador tipo netbook</NombreProd>
      <DescripcionProd>
       Intel Atom, 1600MHz, <b>2GB de memoria RAM</b>,
       disco duro 320 GB, 2 puertos USB, Fast Ethernet.
      </DescripcionProd>
    </Producto>
  </Stock>
  <Facturado totalFacturado="950"/>
</Facturacion>
```

**Nota**: se puede suponer que para todos los productos hay existencias suficientes como para atender a todos los pedidos contenidos en el elemento `Pedidos` del documento fuente.

# Ejercicio 12

La organización de la Eurocopa de fútbol 2012 ha decidido gestionar los datos de la competición (equipos, resultados, etc.) en ficheros XML. Para la primera fase, los datos de cada grupo se almacenan en un fichero XML que se debe ajustar al siguiente DTD:

```
<!ELEMENT GrupoEurocopa (Equipos, Partidos)>
<!ATTLIST GrupoEurocopa
grupoId (A|B|C|D) #REQUIRED>

<!ELEMENT Equipos (Equipo)+>

<!ELEMENT Equipo EMPTY>
<!ATTLIST Equipo
nombre CDATA #REQUIRED
puntos CDATA #IMPLIED
gf     CDATA #IMPLIED
gc     CDATA #IMPLIED>

<!ELEMENT Partidos (Partido)*>

<!ELEMENT Partido (Local, Visitante)>

<!ELEMENT Local (#PCDATA)>
<!ATTLIST Local
equipo CDATA #REQUIRED>

<!ELEMENT Visitante (#PCDATA)>
<!ATTLIST Visitante
equipo CDATA #REQUIRED>
```

El elemento raíz es `GrupoEurocopa`, que contiene un atributo que indica de que grupo se trata (A, B, C o D). Contiene a su vez 2 elementos: `Equipos` (los equipos que componen el grupo) y `Partidos` (los partidos que se juegan en el grupo). Para cada `Equipo` se definen 4 atributos:

- `nombre`: el país que representa el equipo.
- `puntos`: los puntos obtenidos por el equipo en los partidos disputados en el grupo.
- `gf`: goles a favor del equipo en los partidos disputados en el grupo.
- `gc`: goles en contra del equipo en los partidos disputados en el grupo.

Para cada `Partido` se indican los equipos que disputan el partido (elementos `Local` y `Visitante`). Para cada uno de los elementos `Local` y `Visitante` se indica el equipo (atributo `equipo`) y los goles marcados por dicho equipo en el partido (contenido del elemento).

A continuación se presenta un ejemplo de documento XML que se ajusta a este DTD.

```
<?xml version="1.0"?>
<!DOCTYPE GrupoEurocopa SYSTEM "eurocopa.dtd">

<GrupoEurocopa grupoId="B">
  <Equipos>
    <Equipo nombre="Alemania"/>
    <Equipo nombre="Portugal"/>
    <Equipo nombre="Dinamarca"/>
    <Equipo nombre="Holanda"/>
  </Equipos>
  <Partidos>
    <Partido>
      <Local equipo="Holanda">0</Local>
      <Visitante equipo="Dinamarca">1</Visitante>
    </Partido>
    <Partido>
      <Local equipo="Alemania">1</Local>
      <Visitante equipo="Portugal">0</Visitante>
    </Partido>
    <Partido>
      <Local equipo="Dinamarca">2</Local>
      <Visitante equipo="Portugal">3</Visitante>
    </Partido>
    <Partido>
      <Local equipo="Holanda">1</Local>
      <Visitante equipo="Alemania">2</Visitante>
    </Partido>
    <Partido>
      <Local equipo="Dinamarca">1</Local>
      <Visitante equipo="Alemania">2</Visitante>
    </Partido>
    <Partido>
      <Local equipo="Portugal">2</Local>
      <Visitante equipo="Holanda">1</Visitante>
    </Partido>
  </Partidos>
</GrupoEurocopa>
```

Escriba una hoja de estilo XSLT que, dado un documento XML que se ajuste al DTD presentado en el enunciado produzca como resultado un documento XML en el que se ha realizado **exclusivamente** el siguiente cambio respecto del documento XML de entrada:

- A cada elemento `Equipo` del documento XML de entrada se le añaden los atributos `gf` (goles a favor), `gc` (goles en contra) y `puntos`, de acuerdo con los resultados de los partidos contenidos en el elemento `Partidos`.

Dicha hoja de estilo XSLT produciría para el documento XML de ejemplo que se ha presentado más arriba el siguiente resultado:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE GrupoEurocopa SYSTEM "eurocopa.dtd">
<GrupoEurocopa grupoId="B">
<Equipos>
<Equipo nombre="Alemania" gf="5" gc="2" puntos="9"/>
<Equipo nombre="Portugal" gf="5" gc="4" puntos="6"/>
<Equipo nombre="Dinamarca" gf="4" gc="5" puntos="3"/>
<Equipo nombre="Holanda" gf="2" gc="5" puntos="0"/>
</Equipos>
<Partidos>
<Partido>
```

```
<Local equipo="Holanda">0</Local>
<Visitante equipo="Dinamarca">1</Visitante>
</Partido>
<Partido>
<Local equipo="Alemania">1</Local>
<Visitante equipo="Portugal">0</Visitante>
</Partido>
<Partido>
<Local equipo="Dinamarca">2</Local>
<Visitante equipo="Portugal">3</Visitante>
</Partido>
<Partido>
<Local equipo="Holanda">1</Local>
<Visitante equipo="Alemania">2</Visitante>
</Partido>
<Partido>
<Local equipo="Dinamarca">1</Local>
<Visitante equipo="Alemania">2</Visitante>
</Partido>
<Partido>
<Local equipo="Portugal">2</Local>
<Visitante equipo="Holanda">1</Visitante>
</Partido>
</Partidos>
</GrupoEurocopa>
```

Se proporciona la siguiente plantilla para la hoja de estilo XSLT:

```
<?xml version="1.0"?>

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

<xsl:output method="xml" indent="yes" encoding="ISO-8859-1"
            doctype-system="eurocopa.dtd"/>

<xsl:strip-space elements="*"/>

...

</xsl:stylesheet>
```

# Ejercicio 13

Considere el siguiente DTD para definir documentos XML que representan matrices.

```
<!ELEMENT matrixA (filaA+)>

<!ELEMENT filaA (celdaA+)>

<!ELEMENT celdaA EMPTY>

<!ATTLIST celdaA
valor CDATA #REQUIRED>

<!ELEMENT matrixE (filaE+)>

<!ELEMENT filaE (celdaE+)>

<!ELEMENT celdaE (#PCDATA)>
```

A continuación se muestra un documento XML válido de acuerdo con dicho DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE matrixE SYSTEM "matrix.dtd">

<matrixE>
<filaE><celdaE>1</celdaE><celdaE>0</celdaE><celdaE>2</celdaE></filaE>
<filaE><celdaE>3</celdaE><celdaE>1</celdaE><celdaE>5</celdaE></filaE>
</matrixE>
```

APARTADO 1

Como se puede observar, el DTD presentado en este ejercicio permite describir matrices de dos maneras. En las matrices cuyo elemento raíz es `matrixE` el valor de cada elemento de la matriz se indica en el contenido de un elemento `celdaE`, mientras que en las matrices cuyo elemento raíz es `matrixA` el valor de cada elemento de la matriz se indica en el atributo `valor` de un elemento `celdaA`.

Escriba una hoja de estilo XSLT que dado un documento XML cuyo elemento raíz sea `matrixE` y que sea válido respecto del DTD presentado en este ejercicio produzca como resultado un documento XML válido respecto del DTD presentado en este ejercicio y que represente la misma matriz que el documento fuente pero cuyo elemento raíz sea `matrixA`.

Por ejemplo, para el documento XML de ejemplo presentado anteriormente, la hoja de estilo pedida debería producir el siguiente resultado:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE matrixA SYSTEM "matrix.dtd">
<matrixA>
<filaA>
<celdaA valor="1"/>
<celdaA valor="0"/>
<celdaA valor="2"/>
</filaA>
<filaA>
<celdaA valor="3"/>
<celdaA valor="1"/>
<celdaA valor="5"/>
</filaA>
</matrixA>
```

APARTADO 2

Escriba una hoja de estilo XSLT que dado un documento XML cuyo elemento raíz sea `matrixE` y que sea válido respecto del DTD presentado en este ejercicio produzca como resultado un documento XML que sea válido respecto del DTD proporcionado en este ejercicio y que represente el resultado de multiplicar por 2 la matriz representada por el documento fuente.

Por ejemplo, para el documento XML de ejemplo presentado al inicio del ejercicio, la hoja de estilo pedida debería producir el siguiente resultado:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE matrixE SYSTEM "matrix.dtd">
<matrixE>
<filaE>
<celdaE>2</celdaE>
<celdaE>0</celdaE>
<celdaE>4</celdaE>
</filaE>
<filaE>
<celdaE>6</celdaE>
```

<celdaE>2</celdaE>
<celdaE>10</celdaE>
</filaE>
</matrixE>