

Gestão e Tratamento de Informação

1º semestre

Resolução do Mini-Projecto 1 – Grupo 7

Jessica Filipa Assis Alves Ribeiro

Nº 70298

Manuel João Pereira Alves

Nº 70179

Ricardo Manuel Ferreira Leitão

Nº 69632

Solutions for Exercise 1**Question 1.1 – Solution in text file “Parliament.xsd”**

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.parlamento.pt"
elementFormDefault="qualified"
xmlns="http://www.parlamento.pt">
  <xs:element name="parliament" type="parliamentType"/>
  <xs:complexType name="politicianType">
    <xs:simpleContent>
      <xs:extension base="politicianName">
        <xs:attribute name="code" type="xs:integer" use="required"/>
        <xs:attribute name="party" type="xs:string"/>
        <xs:attribute name="age" type="xs:positiveInteger"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:simpleType name="politicianName">
    <xs:restriction base="xs:string">
      <xs:pattern value="([p{Lu}][p{L}]*) ([-p{L}]*)"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="speechType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="order" type="xs:positiveInteger"/>
        <xs:attribute name="politician" type="xs:integer"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="sessionType">
    <xs:sequence>
      <xs:element name="speech" type="speechType" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="date" type="xs:date" use="required"/>
  </xs:complexType>
  <xs:complexType name="parliament-interventionsType">
    <xs:sequence>
      <xs:element name="session" type="sessionType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="politiciansType">
    <xs:sequence>
```

```

        <xs:element name="politician" type="politicianType"
maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="parliamentType">
    <xs:sequence>
        <xs:choice maxOccurs="unbounded">
            <xs:element name="parliament-interventions" type="parliament-
interventionsType"/>
            <xs:element name="politicians" type="politiciansType" />
        </xs:choice>
    </xs:sequence>
</xs:complexType>
</xs:schema>

```

Question 1.2 – Solution in text file “Parliament.xsd”

```

<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.parlamento.pt"
elementFormDefault="qualified"
xmlns="http://www.parlamento.pt">
    <xs:element name="parliament" type="parliamentType">
        <xs:keyref name="politicianCodeKeyRef" refer="politicianCodeKey">
            <xs:selector xpath="parliament-interventions/session/speech"/>
            <xs:field xpath="@politician"/>
        </xs:keyref>
        <xs:key name="politicianCodeKey">
            <xs:selector xpath="politicians/politician"/>
            <xs:field xpath="@code"/>
        </xs:key>
    </xs:element>
    <xs:complexType name="politicianType">
        <xs:simpleContent>
            <xs:extension base="politicianName">
                <xs:attribute name="code" type="xs:integer" use="required"/>
                <xs:attribute name="party" type="xs:string"/>
                <xs:attribute name="age" type="xs:positiveInteger"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:simpleType name="politicianName">
        <xs:restriction base="xs:string">
            <xs:pattern value="([p{Lu}][p{L}]*) ([-'\p{L}]*)"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="speechType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="order" type="xs:positiveInteger"/>
                <xs:attribute name="politician" type="xs:integer"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

```

```

</xs:complexType>
<xs:complexType name="sessionType">
  <xs:sequence>
    <xs:element name="speech" type="speechType" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="date" type="xs:date" use="required"/>
</xs:complexType>
<xs:complexType name="parliament-interventionsType">
  <xs:sequence>
    <xs:element name="session" type="sessionType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="politiciansType">
  <xs:sequence>
    <xs:element name="politician" type="politicianType"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="parliamentType">
  <xs:sequence>
    <xs:choice maxOccurs="unbounded">
      <xs:element name="parliament-interventions" type="parliament-
interventionsType"/>
      <xs:element name="politicians" type="politiciansType" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

Solutions for Exercise 2

Solution in text file "ParliamentToWebTable.xsl"

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:p="http://www.parlamento.pt"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" indent="yes" encoding="ISO-8859-1"/>
  <xsl:template match="/">
    <html>
      <body>
        <xsl:apply-templates />
      </body>
    </html>
  </xsl:template>
  <xsl:template match="*">
    <xsl:apply-templates />
  </xsl:template>
  <xsl:template match="p:politicians">
    <table>
      <tr>
        <td>Code</td>
        <td>Party</td>

```

```

        <td>Age</td>
        <td>Name</td>
        <td>Number of Interventions</td>
        <td>Number of Sessions</td>
    </tr>
    <xsl:apply-templates />
</table>
</xsl:template>
<xsl:template match="p:politician">
    <xsl:param name="code" select="./@code" />
    <xsl:param name="party" select="./@party" />
    <xsl:param name="age" select="./@age" />
    <xsl:param name="name" select="./text()" />
    <xsl:param name="nParlInterv" select="count(//p:parliament-
interventions[./session/speech[@politician = $code]])" />
    <xsl:param name="nSessions" select="count(//p:session[./speech/@politician =
$code])" />
    <tr>
        <td><xsl:value-of select="$code" /></td>
        <td><xsl:value-of select="$party" /></td>
        <td><xsl:value-of select="$age" /></td>
        <td><xsl:value-of select="$name" /></td>
        <td><xsl:value-of select="$nParlInterv" /></td>
        <td><xsl:value-of select="$nSessions" /></td>
    </tr>
</xsl:template>
<xsl:template match="text()" />
</xsl:stylesheet>

```

Solutions for Exercise 3

Question 3.1 – Solution in text file “Exercise 3.1.txt”

```

declare default element namespace "http://www.parlamento.pt";
doc("https://fenix.tecnico.ulisboa.pt/downloadFile/566729524645336/parliamentUdata.xml ")

/count(distinct-values(//parliament-interventions/session[@date = "2014-03-01" or @date = "2014-
03-02"]/speech/@politician))

```

Question 3.2 – Solution in text file “Exercise 3.2.txt”

```

declare default element namespace "http://www.parlamento.pt";
doc("https://fenix.tecnico.ulisboa.pt/downloadFile/566729524645336/parliamentUdata.xml")

//politician[@code = //parliament-interventions/session/speech[text()[(contains(., "ensino superior"))
or text()[(contains(., "educação"))]]/@politician]/text()

```

Question 3.3 – Solution in text file “Exercise 3.3.txt”

```
declare default element namespace "http://www.parlamento.pt";
doc("https://fenix.tecnico.ulisboa.pt/downloadFile/1411154454773835/parliament-data-small.xml")

//politician[@code = //speech[@politician=//politician[text()][starts-with(., "Jos") and ends-
with(., "Seguro")]]/@code]/following::speech[1][@politician !=//politician[text()][starts-with(., "Jos") and
ends-with(., "Seguro")]]/@code ]/@politician]/text()
```

Question 3.4 – Solution in text file “Exercise 3.4.txt”

```
declare default element namespace "http://www.parlamento.pt";
doc("https://fenix.tecnico.ulisboa.pt/downloadFile/566729524645336/parliamentUdata.xml")

/round(/avg(//politician[@code=//session[@date="2014-03-
01"]/speech[@politician=//politician[@party="PSD"]/@code][1]/following-
sibling::*/@politician]/@age))
```

Solutions for Exercise 4

Question 4.1 – Solution in text file “Exercise 4.1.txt”

```
declare namespace p = "http://www.parlamento.pt";

let $doc := doc("file:///afs/ist.utl.pt/users/3/2/ist169632/gti-project1/parliament-data.xml")

let $rep := (
for $speech in $doc//p:speech
let $code := $doc//p:politician[text() = "José Seguro"]/@code
let $replies := $speech[@politician = $code]/following-sibling::*[1][attribute::politician and @politician
!= $code]
order by $replies/@politician
return $replies)

let $codes :=
(let $dist := distinct-values($rep/@politician)
for $i in (1 to count($dist))
let $count := $rep[@politician = $dist[$i]]
return if (count($count) > 2)
then $dist[$i]
else ())

for $code in $codes
let $politician := $doc//p:politician[@code = $code]
return <politician party="{ $politician/@party}">{ $politician/text()}</politician>
```

Question 4.2 – Solution in text file “Exercise 4.2.txt”

```
declare namespace p = "http://www.parlamento.pt";

<interventions>{
let $doc := doc("file:///C:/Users/Ricardo Leitão/Documents/gti-project1/parliament-data.xml")
for $i in (1 to 5)
let $intervention := (for $speech in $doc//p:speech
for $interventions in $speech[attribute::politician]
order by count(tokenize($interventions/text(), "\s")) descending
return $interventions)[$i]
let $politician := $doc//p:politician[@code = $intervention/@politician]
return <intervention session-date="{ $intervention/./@date}"
party="{ $politician/@party}"
politician="{ $politician/text()}"
rank-order="{ $i}">
{ $intervention/text() } </intervention>
}</interventions>
```

Question 4.3 – Solution in text file “Exercise 4.3.txt”

```
declare default element namespace "http://www.parlamento.pt";
let $doc := doc("/Users/jessicaribeiro/Desktop/IST/1semestre/GTI/Projecto/gti-project1/parliament-data-small.xml")

let $sessions := (
for $session in $doc//session
return
copy $s := $session
modify (
insert node ( attribute most-frequent {} ) into $s
return $s
)

let $politicians := (
for $politician in $doc//politician
return
copy $p := $politician
modify (
insert node ( attribute num-interventions {} ) into $p,
insert node ( attribute num-sessions {} ) into $p
return $p
)

let $num-interv := (
for $politician in $politicians
let $count := count($doc//speech[@politician eq $politician/@code])
return
copy $p := $politician
modify (
replace value of node $p/@num-interventions with $count
return $p
)
```

```
let $num-sess := (
  for $politician in $num-interv

  let $interv := (
    for $session in $doc//session
    let $p := distinct-values($session/speech/@politician)
    return $p
  )

  let $code := $politician/@code
  let $count := count(index-of($interv, $code))
  return copy $p := $politician
    modify (
      replace value of node $p/@num-sessions with $count)
    return $p
)

let $m-f := (
  for $session in $sessions
  let $politician := $doc//politician
  let $p := (
    for $pol in $politician
    let $count := count($session/speech[@politician eq $pol/@code])
    return <p order="{ $pol/@code}" num="{ $count}" > </p>
  )
)

let $max := max(
  for $i in $p
  return $i/@num
)

let $politic := $p[@num = $max]

let $party := $politician[@code = $politic/@order]/@party

return
  copy $s := $session
  modify (
    replace value of node $s/@most-frequent with $party)
  return $s)

return
  copy $d := $doc
  modify (
    replace node $d//politicians with <politicians> { $num-sess } </politicians>,
    replace node $d//parliament-interventions with <parliament-interventions> { $m-f }
  </parliament-interventions>
)
return $d
```

Question 4.4 – Solution in text file “Exercise 4.4.txt”

```

declare default element namespace "http://www.parlamento.pt";
let $doc := doc("file:///Users/manuelalves/Desktop/parliament-data.xml")//parliament

let $cpol :=(
for $politician in ($doc//politician)
let $count := count($doc//speech[@politician = $politician/@code])
where $count < 3
return $politician/@code
)

return
copy $d := $doc
modify (
delete node ( $d//politician[@code = $cpol], $d//speech[@politician = $cpol])
)
return $d

```

Question 5

- **Simple Tree Matching Algorithm**
 - **Results of calculating the number of matching nodes**

		name	affiliation	age	occupation
	0	0	0	0	0
name	0	1	1	1	1
party	0	1	1	1	1
age	0	1	1	2	2
occupation	0	1	1	2	8

		52
	0	0
51	0	0

		deputy	party leader	professor
	0	0	0	0
deputy	0	1	1	1
party leader	0	1	5	5
professor	0	1	5	6

		start	end
	0	0	0
start	0	1	1
end	0	1	2

		2011
	0	0
2011	0	0

		first	last
	0	0	0
JOSE SEGURO	0	0	0

○ **Alignment between the trees**

Simple Tree Matching returns Matching = 9, therefore we have 9 matching nodes, including the root. As such, the following tree is created representing the alignment between the trees.

