01: A

```
/employees/employee
/child::employees/child::employee
//employee
//child::employee
/descendant-or-self::node()/child::employee
```

01: B

```
/employees/employee/lastName/text()
/employees/employee/lastName/child::text()
//employee/lastName/text()
//lastName/text()
```

01: C

```
distinct-values(/employees/employee/lastName/text())
```

01: D

```
/employees/employee[lastName = "Smith"]/salary
/employees/employee[lastName/text() = "Smith"]/salary
//employee[lastName = "Smith"]/salary
//employee[child::lastName = "Smith"]/salary
//employee[./child::lastName = "Smith"]/salary
/employees/employee/lastName[. = "Smith"]/../salary
/employees/employee/salary[../lastname = "Smith"]
```

02

```
/employees/employee[salary > avg(/employees/employee/salary)]/email
//employee[salary > avg(//salary)]/email
//employee[./salary > avg(//salary)]/email
//employee[./salary > avg(//salary/text())]/email
//employee[./salary > avg(data(//salary))]/email
```

03

```
data(//department[not(employee)]/@id)
data(//department[not(./employee)]/@id)
data(/departments//department[not(employee)]/@id)
data(//department[count(employee) = 0]/@id)
data(//child::department[not(child::employee)]/attribute::id)
data(/descendant-or-self::node()/child::department[not(employee)]/@id)
```

04: NOT working solution

```
//department[last()]/name
//department[position() = last()]/name
/descendant-or-self::node()/department[last()]/name
```

04

```
/descendant::department[last()]/name
/descendant::department[position() = last()]/name
/descendant-or-self::department[last()]/name
(//department)[last()]/name
(//department[last()])[last()]/name
//department[not(following::department) and not(descendant::department)]/name
//department[not(following::department or descendant::department)]/name
//department[count(following::department) + count(descendant::department) = 0]/name
```

05

```
data(//department[count(.//department) >= 2]/@id)
data(//department[count(./descendant::department) >= 2]/@id)
```

06

```
data(//department[@id = "D1.2.1"]/ancestor-or-self::department[last()]/@id)
data(/departments/department[descendant-or-self::department[@id = "D1.2.1"]]/@id)
```

07

```
for $e in doc("employees.xml")//employee
where $e/lastName != "Smith"
return
   <employee>
     <number>{ data($e/@number) }</number>
     { $e/firstName }
     { $e/lastName }
   </employee>

… where $e/lastName/text() != "Smith" …
… where not($e/lastName = "Smith") …
… where $e/lastName ne "Smith" …
… where not($e/lastName eq "Smith") …

for $e in doc("employees.xml")//employee[lastName != "Smith"]
…

for $e in //employee
let $f := $e/firstName, $l := $e/lastName
…
```

08

```
for $e in doc("employees.xml")//employee
let $n := concat($e/firstName, " ", $e/lastName)
return <employee>{ $n }</employee>

… let $n := concat($e/firstName/text(), " ", $e/lastName/text()) …
… return element employee { text { $n } }
… return element employee { $n }
… return element { "employee" } { text { $n } }

for $e in doc("employees.xml")//employee
return <employee>{ concat($e/firstName, " ", $e/lastName) }</employee>
… return <employee>{ $e/firstName/text() } { " " } { $e/lastName/text() }</employee>
… return element employee { text { concat($e/firstName, " ", $e/lastName) } }
… return element employee { text { $e/firstName }, " ", text { $e/lastName } }
```

Incorrect:

```
… return <employee>{ $e/firstName } { " " } { $e/lastName }</employee>
```

09

```
for $e in doc("employees.xml")//employee
where ($e/salary > 2300) and ($e/worksIn != "D1.1")
order by $e/salary, $e/lastName descending
return text { $e/email }
```

```
… where ($e/salary > 2300) and not($e/worksIn = "D1.1") …
… return text { $e/email/text() }
… return $e/email/text()
```

Incorrect:

```
… return { $e/email/text() }
… return $e/email
```

10

```
for $d in doc("departments.xml")//department
let $c := count($d/descendant::employee)
order by $c
return <department employees="{ $c }">{ $d/@id }{ $d/name/text() }</department>
```

```
… let $c := count($d//employee) …
… order by $c ascending …
… return
     <department employees="{ $c }" id="{ $d/@id }">
       { $d/name/text() }
     </department>
… return
     element department {
       attribute employees { $c },
       attribute id { $d/@id },
       text { $d/name/text() }
     }
… return
     element department {
       attribute employees { $c },
       attribute id { data($d/@id) },
       text { $d/name }
     }
```

Incorrect:

```
… return <department employees="{ $c }">{ $d/@id }{ $d/name }</department>
```

11

```
let $s := doc("departments.xml")//manager
let $x := max(
  for $m in $s
  return count($m/..//employee)
)
for $m in $s
where count($m/..//employee) = $x
return <manager>{ concat($m/firstName, " ", $m/lastName) }</manager>

let $s := doc("departments.xml")//manager
let $i :=
  for $m in $s
  return
    <item employees="{ count($m/..//employee) }">
      { concat($m/firstName, " ", $m/lastName) }
    </item>
let $x := max($i/@employees)
for $m in $i
where $m/@employees = $x
return <manager>{ $m/text() }</manager>

let $s := doc("departments.xml")//manager
let $x := max($s/count(..//employee))
for $m in $s
where count($m/..//employee) = $x
return <manager>{ concat($m/firstName, " ", $m/lastName) }</manager>

let $s := doc("departments.xml")//manager
let $x := max($s/count(..//employee))
for $m in $s/.[count(..//employee) = $x]
where true()
return <manager>{ concat($m/firstName, " ", $m/lastName) }</manager>

let $s := doc("departments.xml")//manager, $x := max($s/count(..//employee)) …
```

12

```
<html>
  <head><title>Departments</title></head>
  <body>
    <table>
    {
      for $d in doc("departments.xml")//department
      let $s := $d/ancestor-or-self::department/manager
      let $m := $s[last()]
      order by $d/@id
      return
        <tr>
          <td>{ data($d/@id) }</td>
          <td>{ $d/name/text() }</td>
          <td>{
                if ($s)
                then concat($m/firstName, " ", $m/lastName)
                else <i>Unknown</i>
          }</td>
        </tr>
    }
    </table>
  </body>
</html>
```