


# <oo> → <dh> Digital humanities

Maintained by: David J. Birnbaum ([djbitt@gmail.com](mailto:djbitt@gmail.com)) 

Last modified: 2014-09-28T20:57:54+0000

## XPath assignment #4

You can find an XML (TEI) version of Shakespeare's *Hamlet* at <http://dh.obdurodon.org/bad-hamlet.xml>. We've deliberately damaged some of the markup in this edition to introduce some inconsistencies, but the file is well-formed XML, which means that you can use XPath to explore it. You should download this file to your computer (typically that means right-clicking on the link and selecting "save as") and open it in <oXygen/>.

After you've completed your homework, save your answers to a file and upload it to CourseWeb as an attachment. (Please use an attachment! If you paste your answer into the text box, CourseWeb may munch the angle brackets.) Some of these tasks are thought-provoking, and even difficult. If you get stuck, do the best you can, and if you can't get a working answer, give the answers you tried and explain where they failed to get the results you wanted. Sometimes doing that will help you figure out what's wrong, and even when it doesn't, it will help us identify the difficult moments. These tasks require the use of path expressions, predicates, and functions. There may be more than one possible answer.

Using the *Bad Hamlet* document and the XPath browser window in <oXygen/>, construct XPath expressions that will do the following (give the full XPath expressions in your answers, and not just the results):

1. What XPath will return a hyphen-separated list of all characters without duplicates. The resulting list will look something like:

Claudius-Hamlet-Polonius ...

Our solution uses **distinct-values()** and **string-join()**. Note that there are several ways to identify the characters in this markup, including the **<castList>** element, the **<speaker>** elements, and the **@who** attribute on the **<sp>** element. Which should you use and why?

2. Most metrical lines (**<l>**) have an **@xml:id** attribute with a value like "sha-ham101010", ending in a six-digit number. The first digit is the act, the next two the scene, and the last three the line in the scene. Some metrical lines are split across multiple speakers, and in that case the six-digit number in the **@xml:id** value is followed by "I" (initial part), "M" (middle part), or "F" (final part). In a few places there may be more than one middle part, and in those cases the "M" is followed by a one-digit number. For example, one of Hamlet's lines is:

<l xml:id="sha-ham502277M2" n="277">One.</l>

which is the second middle part. What XPath will return the number of **<l>** elements that are middle parts? Our solution uses **count()** and **contains()**.

3. Sometimes Rosencrantz speaks by himself and sometimes he speaks in unison with Guildenstern.
  - a. What XPath finds all of the speeches by Rosencrantz, whether alone or together with Guildenstern? Our solution uses a single instance of **contains()**.
  - b. Can you think of an alternative solution that doesn't use any functions (just a predicate)?
4. The **string-length()** function can be used in two ways. You can wrap it around an argument, so that, for example, **string-length('Hi, Mom!')** will return 8, the length in character count of the string inside the quotation marks. It can also be used as part of a path expression, so that, for example, if the XPath **//sp** returns a sequence of all **<sp>** elements, **//sp/string-length(.)** returns a sequence of the lengths of all **<sp>** elements as measured by counting characters. This works by finding all of the **<sp>** elements and then (next path step) getting the string length of each one. Remember that the dot inside the parentheses refers to the current context node, which is the member of the sequence of **<sp>** nodes that is being processed at the moment. We need to use the subterfuge because **string-length(//sp)** generates an error. The problem is that **string-length()** can take only a single argument, and **//sp** returns more than one item. Putting the **string-length()** function on its own path step with a dot inside means that it applies once for every **<sp>** element, and that each time it applies, it has just a single argument.

Use this information to identify an XPath that finds the length of the longest speech. What length does it return? Our solution uses **string-length()** and **max()**.
5. *Optional, challenging question:* Given the preceding solution, how can you use that XPath to retrieve the longest **<sp>** itself? No fair checking the length and then writing a separate XPath that looks for that number. Your answer must find the longest speech without your knowing how long it is. Our solution doesn't require any additional functions beyond the ones used in #4, but it does use a complicated predicate.
6. *Optional, very challenging question:* What XPath produces a numbered list of all characters, without any duplicates, which should look something like:
  1. Claudius
  2. Hamlet
  3. Polonius
  4. ...

There are several possible solutions, each of which raises issues that you may not have seen before. If you get an error message, try to figure out what it means and how to resolve it.