Pandoc filter exercises

1. Write a filter that capitalizes all regular text in a document, leaving alone URLs and link titles.
2. Write a filter that promotes all level 2 headers to level 1, level 3 to level 2, etc.
3. Write a filter that removes all horizontal rules from a document.
4. Write a filter that converts strong emphasis to underlining. The filter should work for both HTML and LaTeX.

   Hints: Pandoc has no `Underline` inline element. But it does have a `RawInline` inline element, which you can use to directly specify what the output should look like in a particular format. You'll need to make the filter sensitive to the output format.
5. Write a filter that converts all links in a document into regular text followed by the URL in parentheses.
6. Create a filter that finds code blocks with attribute `include="FILENAME"`, that is,

   ~~~ {include="myprog.lua"}
   contents go here...
   ~~~

   and replaces their contents with the contents of FILENAME.
7. Create a filter that finds all code blocks with class `python`,

   ``` python
   # python code here!
   ```

   runs the python interpreter on their contents, and appends a separate code block (with class `output`) containing the output produced.
8. Provide a simple markdown way to write ruby markup, a method of providing phonetic transliterations above Chinese and Japanese characters:

   これは紙芝居(かみしばい)です。

   ```
   これは<ruby>
     <rb>紙芝居</rb>
     <rt>かみしばい</rt>
   </ruby>です。
   ```
9. For LaTeX users: Create a filter that allows you to use inline tikz diagrams in markdown. Hint 1: They already work with LaTeX output, if you include `\usepackage{tikz}` in your template. Make them work with HTML, including HTML slide show formats and epub. Hint 2: You can use pdflatex and ImageMagick to generate the images. Read up on `\documentclass{standalone}`.

Code examples

1. `AllCaps.hs`

   ```haskell
   module AllCaps (allCaps) where
   import Text.Pandoc.Definition
   import Data.Char (toUpper)

   allCaps :: Inline -> Inline
   allCaps (Str xs) = Str $ map toUpper xs
   allCaps x = x
   ```

2. `caps.hs`: run with `pandoc --filter ./caps.hs`

   ```haskell
   import Text.Pandoc.JSON
   import AllCaps (allCaps)

   main = toJSONFilter allCaps
   ```

3. `emphToCaps.hs`

   ```haskell
   import Text.Pandoc.JSON
   import Text.Pandoc.Walk
   import AllCaps (allCaps)

   emphToCaps :: Inline -> [Inline]
   emphToCaps (Emph xs) = walk allCaps xs
   emphToCaps x = [x]

   main :: IO ()
   main = toJSONFilter emphToCaps
   ```

4. `emphToCaps2.hs`

   ```haskell
   import Text.Pandoc.JSON
   import Text.Pandoc.Walk
   import AllCaps (allCaps)

   emphToCaps :: Maybe Format -> Inline -> [Inline]
   emphToCaps (Just f) (Emph xs)
     | f == Format "html" || f == Format "latex" = [SmallCaps xs]
   emphToCaps _ (Emph xs) = walk allCaps xs
   emphToCaps _ x = [x]

   main :: IO ()
   main = toJSONFilter emphToCaps
   ```