Assignment #1

SUNDAY, JANUARY 28TH, 2018

RANDY DO

Contents

Problem 1

```
1.  Demonstrate that you know how to use "curl" well enough to
correctly POST data to a form.  Show that the HTML response that
is returned is "correct".  That is, the server should take the
arguments you POSTed and build a response accordingly.  Save the
HTML response to a file and then view that file in a browser and
take a screen shot.
```

SOLUTION

The Solution for this problem is:

1)

For this problem, I used Windows. The website was provided by instructor:
http://cs.odu.edu/~anwala/files/temp/namesEcho.php

2) The approach is using the curl commands to POST. We know the variables for POST: fname, lname. We also, need the headers information too.

3) For this, I used -F (specify HTTP multipart POST data) and -i (include protocol headers in the output) *X POST isn't required, but I used it just in case.

```
C:\Users\Randy>curl -i -X POST -F "fname=Randy" -F "lname=Do" http://www.cs.odu.edu/~anwala/files/temp/namesEcho.php
```

4)

Command Prompt

```
C:\Users\Randy>curl -i -X POST -F "fname=Randy" -F "lname=Do" http://www.cs.odu.edu/~anwala/files/temp/namesEcho.php
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Server: nginx
Date: Sat, 20 Jan 2018 22:56:45 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding

<!DOCTYPE html>
<html>
<body>

<br />
<br />
<b>fname Posted: </b>Randy<br />
<b>lname Posted: </b>Do<br />

</body>
</html>
C:\Users\Randy>
```

5) Now that we have the html code from using curl, I copy and paste from the <!DOCTYPE html> to </html> from cmd to a html file.

6) The result by opening from the browser is:

fname Posted: Randy
lname Posted: Do

Problem 2

```
2.  Write a Python program that:
  1. takes as a command line argument a web page
  2. extracts all the links from the page
  3. lists all the links that result in PDF files, and prints out
     the bytes for each of the links.  (note: be sure to follow
     all the redirects until the link terminates with a "200 OK".)
  4. show that the program works on 3 different URIs, one of which
     needs to be:
     http://www.cs.odu.edu/~mln/teaching/cs532-s17/test/pdfs.html
```

SOLUTION

1) `takes as a command line argument a web page`

1a) First, I installed and imported, **requests, sys, BeautifulSoup**

```
import requests
import sys
from bs4 import BeautifulSoup
```

1b) In the python code:

```
website= sys.argv[1]
url = website
r  = requests.get(url)
data = r.text
soup = BeautifulSoup(data)
```

1c) The input should look like this:

```
python hw1.py http://www.cs.odu.edu/~mln/teaching/cs532-s17/test/pdfs.html
```

2) `extracts all the links from the page`

```
for link in soup.find_all('a'):
    print(link.get('href'))
```

Using **BeautifulSoup**, we can get all the links coming from 'href'

3) lists all the links that result in PDF files, and prints out
the bytes for each of the links. (note: be sure to follow
all the redirects until the link terminates with a "200 OK".)

3a) To approach this, I used **requests.** This will help me get the bytes of the pdf files (Content-Length)

```
if link['href'].endswith('.pdf'):
    res = requests.head(link['href'])
    res.headers
    {'content-length': ''}
```

3c) Since we are looking for links, I searched for 'href' that starts with http: or https:

```
link['href'].startswith('https:') or link['href'].startswith('http:')
```

3c) When I used the python code on a Wikipedia page, I would get a KeyError issue. Since, the goal for this is to find links for each website, I used **try:** The python code, **except KeyError:** pass

```
import requests
import sys
from bs4 import BeautifulSoup

# main
website= sys.argv[1]
url = website
r  = requests.get(url)
data = r.text
soup = BeautifulSoup(data)

for link in soup.find_all('a'):
    try:
        if link['href'].startswith('https:') or link['href'].startswith('http:') :
            if link['href'].endswith('.pdf'):
                res = requests.head(link['href'])
                res.headers
                {'content-length': ''}
                print (link.get('href'), res.headers['content-length'], "Bytes")
            else:
                print(link.get('href'))
    except KeyError:
        pass
```

4) Show that the program works on 3 different URIs, one of which needs to be:
http://www.cs.odu.edu/~mln/teaching/cs532-s17/test/pdfs.html

http://www.cs.odu.edu/~mln/teaching/cs532-s17/test/pdfs.html

```
http://twitter.com/webscidl
http://www.dlib.org/dlib/november15/vandesompel/11vandesompel.html
http://arxiv.org/abs/1508.02315
http://arxiv.org/abs/1508.02315
http://www.cs.odu.edu/~mln/pubs/ht-2015/hypertext-2015-temporal-violations.pdf 2184076 Bytes
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-annotations.pdf 622981 Bytes
http://arxiv.org/pdf/1512.06195
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-off-topic.pdf 4308768 Bytes
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-stories.pdf 1274604 Bytes
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-profiling.pdf 639001 Bytes
http://dx.doi.org/10.1007/s00799-015-0150-6
http://www.cs.odu.edu/~mln/pubs/jcdl-2014/jcdl-2014-brunelle-damage.pdf 2205546 Bytes
http://arxiv.org/abs/1506.06279
http://dx.doi.org/10.1007/s00799-015-0155-1
http://bit.ly/1ZDatNK
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-mink.pdf 1254605 Bytes
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-arabic-sites.pdf 709420 Bytes
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-dictionary.pdf 2350603 Bytes
http://bit.ly/jcdl-pdf
http://dx.doi.org/10.1007/s00799-015-0140-8
```

https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

The output is too large for this document (link is provided)

https://ibb.co/frD4zG

https://en.wikipedia.org/wiki/Rho_Ophiuchi

```
http://simbad.u-strasbg.fr/simbad/sim-id?Ident=rho+Oph
http://simbad.u-strasbg.fr/simbad/sim-id?Ident=rho+Oph+A
http://simbad.u-strasbg.fr/simbad/sim-id?Ident=rho+Oph+B
http://simbad.u-strasbg.fr/simbad/sim-id?Ident=rho+Oph+C
http://simbad.u-strasbg.fr/simbad/sim-id?Ident=rho+Oph+D
http://adsabs.harvard.edu/abs/2007A&A...474..653V
http://adsabs.harvard.edu/abs/2002yCat.2237....0D
http://adsabs.harvard.edu/abs/1988MSS...C04....0H
http://adsabs.harvard.edu/abs/1978A&AS...34....1N
http://adsabs.harvard.edu/abs/2006AstL...32..759G
http://adsabs.harvard.edu/abs/2003AN....324..219W
http://adsabs.harvard.edu/abs/2007BaltA..16..435N
http://adsabs.harvard.edu/abs/2013ApJ...764L..10C
http://adsabs.harvard.edu/abs/2008hsf2.book..351W
http://adsabs.harvard.edu/abs/2002A&A...382...92S
https://en.wikipedia.org/w/index.php?title=Rho_Ophiuchi&oldid=822821896
https://donate.wikimedia.org/wiki/Special:FundraiserRedirector?utm_source=donate&utm_medium=sidebar&utm_campaign=C13_en.wikipedia.org&uselang=en
https://www.wikidata.org/wiki/Special:EntityPage/Q782342
https://af.wikipedia.org/wiki/Rho_Ophiuchi
https://ast.wikipedia.org/wiki/Rho_Ophiuchi
https://bg.wikipedia.org/wiki/%D0%A0%D0%BE_%D0%97%D0%BC%D0%B8%D0%B5%D0%BD%D0%BE%D1%81%D0%B5%D1%86
https://es.wikipedia.org/wiki/Rho_Ophiuchi
https://fa.wikipedia.org/wiki/%D8%B1%D9%88_%D9%85%D8%A7%D8%B1%D8%A7%D9%81%D8%B3%D8%A7%DB%8C
https://it.wikipedia.org/wiki/Rho_Ophiuchi
https://pl.wikipedia.org/wiki/Ro_Ophiuchi
https://pt.wikipedia.org/wiki/Rho_Ophiuchi
https://ru.wikipedia.org/wiki/%D0%A0%D0%BE_%D0%97%D0%BC%D0%B5%D0%B5%D0%BD%D0%BE%D1%81%D1%86%D0%B0
https://sk.wikipedia.org/wiki/R%C3%B3_Ophiuchi
https://fi.wikipedia.org/wiki/Rho_Ophiuchi
https://sv.wikipedia.org/wiki/Rho_Ophiuchi
https://zh.wikipedia.org/wiki/%E5%BF%83%E5%AE%BF%E5%A2%9E%E5%9B%9B
https://www.wikidata.org/wiki/Special:EntityPage/Q782342#sitelinks-wikipedia
https://wikimediafoundation.org/wiki/Privacy_policy
https://www.mediawiki.org/wiki/Special:MyLanguage/How_to_contribute
https://wikimediafoundation.org/wiki/Cookie_statement
https://wikimediafoundation.org/
```

Problem 3

```
Consider the "bow-tie" graph in the Broder et al. paper (fig 9):
    http://www9.org/w9cdrom/160/160.html

    Now consider the following graph:

    A --> B
    B --> C
    C --> D
    C --> A
    C --> G
    E --> F
    G --> C
    G --> H
    I --> H
    I --> K
    L --> D
    M --> A
    M --> N
    N --> D
    O --> A
    P --> G

    For the above graph, give the values for:

    IN:
    SCC:
    OUT:
    Tendrils:
    Tubes:
    Disconnected:
```

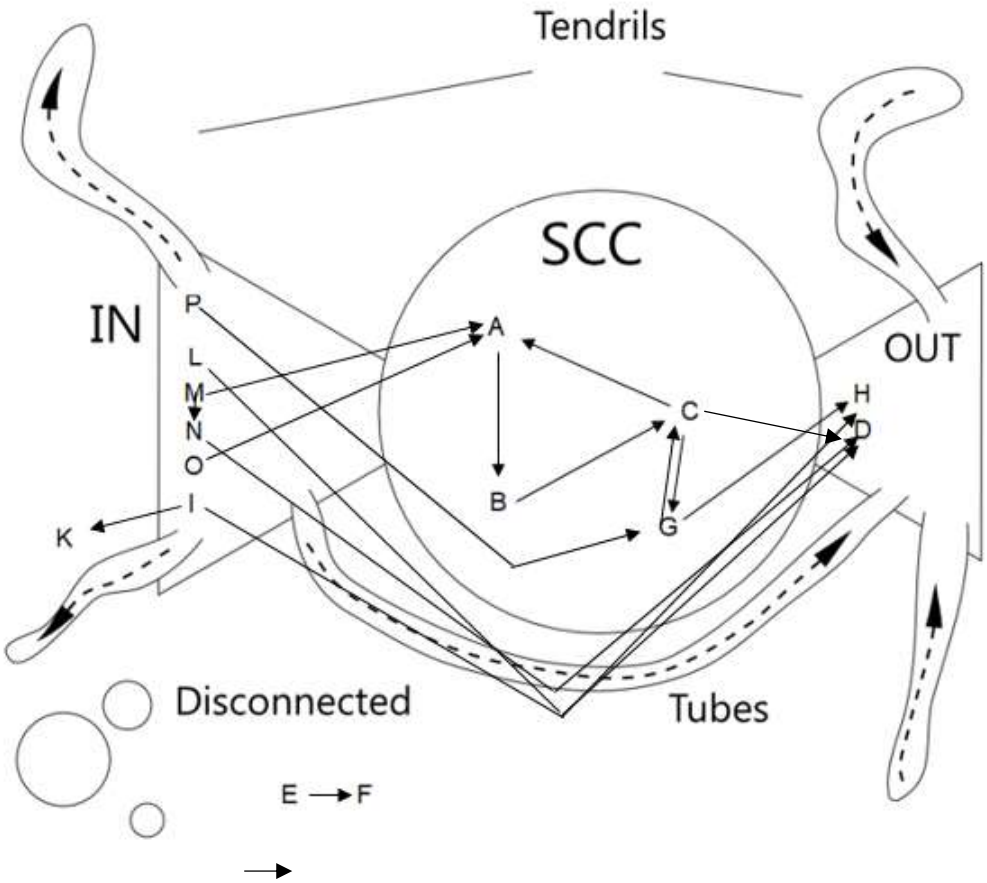
IN: (Pages with no in-links or in-links from IN pages) and out-links to pages in IN, SCC, Tendrils, or Tubes.

OUT: (Pages with no out-links or out-links to other pages in OUT), and all in-links come from OUT, SCC, Tendrils, or Tubes

TENDRILS: (Pages that can only be reached from IN) or have only out-links to OUT

TUBES: (Pages that have in-links from IN or other pages in Tubes) and out-links to pages in Tubes or OUT

DISCONNECTED: (Pages that have no in-links from any other components) and no out-links to other components. These pages may be linked to each other



IN: P, L, M, N, O, I

SCC: A, B, C, D, G

OUT: H. D

Tendrils: K,

Tubes: L, I, N

Disconnected: E, F