Server-Side Web Programming
Assignment 01

Questions

1. Use `nslookup` to determine the IP addresses for the following Internet hosts:

   a. `www.cs.trincoll.edu`
   b. `mail.trincoll.edu`
   c. `www.facebook.com`

   Explain the output of the `nslookup` command and the similarities and differences between each of the hosts. What can you say about the similarities of the IP addresses for a. and b.?

2. What role does Transport Control Protocol (TCP) and Internet Protocol (IP) play in Internet communication?
3. How does HTTP and TCP relate to a telephone call?
4. User Datagram Protocol (UDP) is used for specific kinds of applications. Please describe the type of application that uses UDP and provide explicit examples of applications that might use UDP.
5. What is the difference between the header and body of an HTML request?
6. Use `telnet` to send an HTTP request using telnet to the following Internet hosts:

   a. `www.cs.trincoll.edu`
   b. `mail.trincoll.edu`
   c. `www.mit.edu`

   Do not forget to include the Host header field. Explain ***why*** you might also want to include the `Connection: close` header in your requests. For each response, list and describe the header fields that each host returns. Which HTTP command can you use to only return the HTTP response header?

7. For each of the hosts in 5. give a list of the HTTP methods that are allowed by each.
8. How can a web browser retrieve an HTML document from a web server if DNS is broken?
9. What is a request-response protocol? Given an example.
10. What is a stateless protocol? Is HTTP stateful or stateless? Please explain.

Programming Exercise

A *web spider* or *bot* is a program that accesses web documents automatically rather than through a web browser. These programs are typically used to index web sites for searching purposes. For example, the Google search engine uses a program called googlebot to automatically crawl the web and build a searchable index of web pages. An indexing bot begins with a starting web document, reads all the links contained in that document, and then follows each of those links to new pages, and recursively repeats this process for each of the new pages. Some sites disallow bots from reading certain documents[1].

In this programming exercise, you need to extend a simple bot program written in Java so that it can successfully retrieve web documents from the Internet. The source code contains three Java files:

`WebBot.java`	This is the main entry point to the program.
`LinkExtractor.java`	This class is used to extract URLs from an HTML file. **You do not need to modify this file!**
`Index.java`	This class is used to index URLs and the pages they refer to.

[1] http://www.robotstxt.org/orig.html

You must implement two methods to make the web bot functional. The first method is the `crawl` method contained in the `WebBot` class:

```
/**
 * Crawls the web starting at the start URL and returns
 * an Index object of the URLs that were successfully
 * visited.
 *
 * @return an Index
 */
public Index crawl() {
  // TODO
  // 1. Create an Index object to index the URLs.
  // 2. Loop from 0 to MAXURL:
  //      a. Call the poll method on the vqueue object to get
  //         the next URL.
  //      b. If the URL is null, "break" out of the loop.
  //      c. Otherwise, create a LinkExtractor object and
  //         extract the links from the URL.
  //      d. Next, index the link with the Index object.
  //      d. Lastly, loop over all the URLs that are returned by
  //         the link extractor and call the offer method on
  //         the vqueue object to insert the URL in the queue.
  Index idx = new Index();

  // START YOUR CODE

  // END YOUR CODE

  return idx;
}
```

The second method is the `index` method contained in the `Index` class:

```
/**
 * Indexes a URL.
 *
 * @param url the url to index
 */
public void index(URL url) {
  // TODO
  // 1. Create an InputStreamReader object called rdr to wrap the InputStream
  //    that is returned from url.openStream().
  // 2. Create a BufferedReader object called bdr wrapping the rdr object.
  // 3. Read each line of the bdr object by calling the readLine method
  //    until a null is returned.  Capture the lines that are read into a
  //    String called page.
  // 4. Lastly, use the System.out.println method to print the string,
  //    ("indexed " + url).
  //
  // Note: make sure you deal with the IOException that is thrown by the
  //       readLine method.
}
```

Each method includes comments that guide you toward the solution. Here are some links to some useful resources that you may want to consult during your implementation:

http://github.com/timdrichards/webbot
This is the remote git repository where you can access this programming exercise.

http://download.oracle.com/javase/6/docs/api/java/util/Queue.html
This is the documentation for the Queue interface that is used in the WebBot class.

http://www.developer.com/java/other/article.php/3343771/Using-Foreach-Loops-in-J2SE-15.htm
A useful article describing the Java 1.5 for-each loops.

http://download.oracle.com/javase/6/docs/api
A link to the Java 1.6 API documentation. This is useful for all the classes in the JDK. For this assignment you will be interested in looking at the documentation for InputStreamReader, InputStream, and BufferedReader.

Points

Each question in the Questions Section are worth 5 points for a total of 50 points for that section. The programming exercise is also worth 50 points. You will be graded on the completeness of your answers and the accuracy of your response. Each answer must be typed and clearly labelled as to which question your answer corresponds to. *Please read each question carefully!* The programming exercise is graded on the correctness of your solution. You will be severely penalized if your program does not compile. Partial credit is given, so please add comments to your code to show me your thought process. Also, make sure you commit your changes to git along the way so I can see your progression through out the assignment.

Submission

You are to submit your program as a zip archive file and attached to an email. If you are not sure how to do this on the platform you are using please ask me. Here is how you do it in Linux:

```
$ zip -r richards-asn01 webbot
```

Your zip archive should include your solution to the questions and the name of the zip file must be `<your name>-asn01.zip`.