CSE 303, Spring 2005, Assignment 7
Due: Friday 3 June, 9:00AM

Last updated: May 24

Summary: You will develop a CGI program providing a web interface to your solution to homework 6. Your program will use your homework 6 solution, a small Java program for processing inputs, and basic Linux utility programs. The course website provides a mostly-complete solution; you need to add some functionality and change some behavior, as described below.

Remember that the department allows CGI programs only on the machine `abstract` (in the `www` subdirectory of your `abstract` home directory) and that your `attu` home directory is (thankfully) not available from this machine. You will need to transfer files to `abstract`.

1. HTML
   (a) Copy the HTML source in `http://abstract.cs.washington.edu/~djg/303hw7.html` to `http://abstract.cs.washington.edu/~userid/303hw7.html` where `userid` is your userid.
   (b) Change your copy of `303hw7.html` so that the phrase "CSE303 Spring 2005" is hyperlinked to the course homepage.
   (c) Change `303hw7.html` so that the words in the phrase can have up to 35 characters instead of 30. Change the text to indicate the change.
2. Java
   (a) Copy the file `hw7.java` from the course web-site. It expects a query-string as its one and only command-line argument. It allows the form fields to appear in *any order*. (You can use your HTML file to see what the input string looks like; it's the part after the `?` in the URL.)
   (b) Change the method `check_string` so that it returns true if and only if its argument contains only English letters and hyphens.
   (c) Change the method `main` so that the program prints `a b c d` where `a` is the name of the file to be processed and `b c d` is the phrase to look for. If there is *any* problem with the input, it should output *nothing*. The lines you need to change are indicated in the file. In particular you need to:
      - Set the `names` array to hold the correct "field names" for processing the input.
      - Assign `starts[4]` so that the following loop works correctly.
      - Add code to check if all 4 outputs (referred to above as `a`, `b`, `c`, `d`) were correctly figured out.
      - Add code that prints out the 4 outputs.
3. C-Shell
   (a) Copy the file `hw7.cgi` from the course web-site.
   (b) Change `hw7.cgi` so that it sets the variable `myargs` to the standard out produced by calling your Java program with the contents of the environment variable `$QUERY_STRING`.
   (c) Change `hw7.cgi` so that the first argument to `./phrase_chance` is a filename for a file in the directory `~djg/www`. For example, instead of passing the argument `swanh`, you want to pass `~djg/www/swanh`. (At this point, your program "should work".)
   (d) Change `hw7.cgi` so the first line of output includes the first-line of the input-file *in quotation marks* instead of the input-file's name. For example, instead of `Text:  swanh`, output

```
Text: " STAR WARS Episode IV A NEW HOPE "
```

   Hint: `man head`.
   (e) Change your copy of `hw7.cgi` so that the HTML it produces includes a link back to your `303hw7.html`.

Extra Credit 1: Add a second form to your web page. The form should take one word instead of 3 and use your solution to homework 6 to compute how many times that word appears in the document.

- The only programs you may use to process the input file are `wc` and `phrase_chance`.
- Here's the trick: If the word is w and the total number of words in the file is n, then the answer is $(m \cdot n^3)^{1/3}$ where m is the one-word model prediction of seeing the phrase $w\ w\ w$.
- Write new C-Shell and Java files as necessary. (Turn in a tarball with them all.)
- Round the floating-point result to the nearest integer.

Extra Credit 2: Turn in code for a third application `phrase_chance_extra2`. Add a third form to your web page. The form should replace the radio buttons with a text box accepting up to 30,000 characters. You should treat the contents of this text box as the input file for `phrase_chance`. Be sure not to leave around any temporary files you might create.

- You may want to use the "post" method of running CGI programs.
- Write new C-Shell and Java files as necessary. (Turn in a tarball with them all.)

Permissions and Academic Integrity: As usual, you should make your solutions unreadable by other users and may not look at others' solutions even if they set their permissions incorrectly. However, this policy will not work for the HTML file because the whole point is to make it available to the world. So it's worth emphasizing that even though every student's `303hw7.html` file will be available, you may not look at another student's HTML source.

Turn-in Instructions:

- Turn in your HTML, CGI, and Java files using the link on the course webpage. You do not need to turn in the source code for Homework 6.
- Make sure your program is publicly accessible on the web via `http://abstract.cs.washington.edu/~userid/303hw7.html` where `userid` is your userid.