Homework 2

Due Friday, October 14

Here you will develop a PDF indexing tool. You can use it to create an index for, and then browse through, any searchable PDF file. Here are the specs:

- This will be a standalone program, as opposed to just a function. It will be named **index.py**, with usage

  ```
  % python index.py PDF_file_name index_file_name [word_file_name]
  ```

  with the brackets indicating an optional command line argument.

- If the third command line argument is present, the program generates an index of words in that file, listing their page numbers within the PDF file. (Needless to say, this file is user-supplied.) The index is then saved to disk under the file name specified in the second argument. Otherwise, the index file in the latter argument is read in.

- The format of the words file is one word per line, in the form of the word and its variants, separated by slashes, e.g.

  ```
  go/gone/goes/going/went
  ```

  (It would be nice to allow phrases rather than just words, but we'll stick to words.)

- If an index is to be created, your program will run **pdftotext** to produce a **.txt** file, and will generate its index from there. Note that this means that the specific output format of **pdftotext** is assumed; the program **/usr//bin/pdftotext** on CSIF defines our standard.

- It is assumed that the first page in the PDF document is Page 1. Thus for instance it is assumed that there is no Roman-numbered front matter.

- Words are defined as anything separated by white space. Punctuation and non-ASCII characters are removed first. The indexing is not case-sensitive.

- The index file is formatted one word per line, with page numbers separated by commas, e.g.

  ```
  call/calls/called/calling 22,25,30,31,66
  loops 3,29,30,31,52
  ```

  There are no duplicate page numbers within a line.

- The display will use **curses**, a chapter on which is in our textbook. (This material will not be covered in class.) The screen will be split in two subwindows, the upper half showing part of the PDF file (in its **.txt** form), and the lower half showing part of the index. (There is machinery on the Web you can use to set up the two subwindows, or you can just have your own code manage it.) Refer to the two halves as the source window SW and the index window IW.

- The available user commands are (user does NOT hit the Enter key):

  - **o:** Move to the other subwindow. (Set the initial subwindow to the index one.)

  - **u, d:** Move the cursor one line up or down line in the current subwindow, scrolling if

need be (see below).

- **l, r:** Move the cursor one character position left or right within a line. Mainly relevant for IW (see below).
- **v:** View next instance of the current index word (see below).
- **q:** Quit.

- When the cursor is in a line in the IW, the associated word is considered the current word (CW), even if you move the cursor to the SW.

  Think of the page numbers for a given word to be links to the actual pages, viewable in the SW. When you move the cursor to some line, so that CW is set, you can follow various links for that word, using the **v** command. You do this either by moving the cursor to a place within one of the page numbers and then issuing the **v** command, or by issuing the command *without* first moving the cursor. In the latter case, it will be assumed that you are interested in the link immediately after the last one you pursued for that word. Once you move to a different word, there is no default for **v** until you issue a **v** command on a specific link by moving the cursor there.

  For example, say we move the cursor to somewhere in the index line for *loops* above. We might move the cursor to the 29 (either on the 2 or the 9). If we give the **v** command, the program will move the SW to that page. If we then issue the **v** command again, it will move the SW to page 30. If we then decide we want to see the instance of *loops* on page 3, we can move the cursor back to the 3 in the IW. After that, if we move the cursor to the 22 in the *call* line and then hit **v**, the SW goes to page 22.

- When you pursue a link and the SW goes to that page, the program will highlight all instances of the given word in the currently-visible part of the document. The highlighting can be reverse-video, color or whatever.
- Scrolling will occur when you "go off the end of the Earth" (you decide how much to scroll, as long as the desired line does appear).

I have written a debugging interface for R, using Python **curses**, which I had planned to make available to you anyway, for the R portion of the course. But I'll try to give it to you at least in rough form within a day or two of posting this homework.

Extra Credit:

Originally I wanted the user to be able to click the mouse on a page number in the index, and then have the SW move to that page. In theory, this should work, but somehow I haven't succeeded in it. The example here does work both on my home and office Ubuntu machines, and my own test code works on my Ubuntu machine at home, but not on my Ubuntu machine at the office. But you are probably more resourceful than I, and should be able to make it work. If you do, you need not implement the **v** command, and you will get Extra Credit.