Overview
In this project, you will gain experience using PHP to read records from a MySQL database and then display them to a webpage by outputting HTML.

What to do
For this project, do the following:

- In the MySQL account and database that was assigned to you on pearl.ils.unc.edu (e.g., webdb_1), create a table called "p2records" with the following fields and data types:

itemnum	int
authors	varchar(200)
title	varchar(200)
publication	varchar(100)
year	char(4)
type	varchar(20)
url	varchar(255)

- Use the MySQL bulk loader (as described in class) to load records into the p2records table from the file called `bigrecords.txt` that is posted in the Resources section of the Sakai site. The format of the bigrecords.txt file will be one record per line with fields separated by vertical bars:

```
itemnum|authors|title|publication|year|type|url
```

  Each record is guaranteed to have seven fields as shown above. There will be no leading or trailing spaces. All data will be valid – you do not need to worry about checking the values. Itemnum will be a unique value for each record that can be used as the primary key for the p2records table. There is also a file called smallrecords.txt (also in the Sakai Resources) that contains records in the same format that you may find useful for testing.

- Write PHP code to query the p2records table in your database and then display the records in an HTML table with the following headings: authors, title, publication, year, type. The title should be a hyperlink to the url for that record so that if you click on the title in a web browser, it should take you to the page specified by the URL.

- By default, your PHP code should display the records in order by item number (itemnum). However, users should be able to click any of the column headings to re-sort the records by that field. This means that information will need to be transferred from the client to the server when users click one of the column headings. There are several ways to handle this, but one simple way is to have the column headers be <a href> links that contain fields that will be sent to the PHP script using the GET method. In other words, if you have links like this:

```
<a href="browse.php?sortby=author>Author(s)</a>
<a href="browse.php?sortby=title>Title</a>
```

  your PHP code should be able to read the "sortby" variable from the $_GET superglobal.

- In addition to the functionality described so far, implement a simple paging mechanism. Your PHP code should display 25 records at a time. At the bottom of the page, include links that allow the user to go forward or backward in through the records in increments of 25 records. The exact

details of this interface are left to you to decide, but the links should not cause errors (e.g. a user should not be allowed to go "back" if they are looking at records 1-25).

Sorting and paging should be able to be used together and sorting should be done before pages are determined. For example, a user should be able to view titles that start with letters at the end of the alphabet by going to the last page of results while sorting by title.

- Here is a (possibly useful) hint about how to let MySQL help with the sorting and paging:

```
SELECT * FROM p2records ORDER BY title LIMIT 100, 25
```

How to do it
Implement the functionality described above using PHP. Be sure that your files work on ruby.ils.unc.edu with Firefox.

Develop your solution using PHP, HTML, and CSS files as needed. **The main PHP file of your application should be named youronyen-p2-browse.php**. Replace `youronyen` with your Onyen. If you use other files, name them with a prefix of: `youronyen-p2-`
For example:

```
rcapra-p2-browse.php
rcapra-p2-styles.css
```

All your code and links should work with these files, named as indicated, in the same directory. In other words, if your files were placed together in some other directory, they should still work.

Note that for this and several future assignments, you will turn-in: (1) all the files you created, packaged into a tar file (see below), and (2) you will also need to submit a URL to a "live" version of your project running on ruby.ils.unc.edu. I will post an announcement to the Sakai site that contains important information about part (2). Make sure your read and understand the announcement and if you have any questions, feel free to contact me.

In your tar file, include a PLAIN TEXT file called `youronyen-p2-readme` that contains ONLY the URL for your "live" project, followed by a blank line. This URL should start with:

```
http://www.ils.unc.edu/~youronyen/
```

(replace `youronyen` with your Onyen). Do not include any text other than the URL (e.g. no html) and the blank line.

What to turn in and how
Make sure all of your files are in one directory on a Unix machine (for example, ruby.ils.unc.edu). `cd` into that directory and then type the following command to create a gzipped tar file:

```
tar cvzf youronyen-p2.tgz .
```

Note that after the "tgz" there is a space and then a period.
Check your tgz file with the following command:

```
tar tvzf youronyen-p2.tgz
```

If all is well, you should see a display similar to the one below.

```
[rcapra@ruby]$ tar tvzf rcapra-p2.tgz
```

```
drwx------ rcapra/users      0 2007-01-17 16:57:29 ./
-rw------- rcapra/users      0 2007-01-17 16:57:29 ./rcapra-p2.tgz
-rw------- rcapra/users   4343 2007-01-17 16:48:16 ./rcapra-p2-browse.html
-rw------- rcapra/users   4343 2007-01-17 16:48:16 ./rcapra-p2-readme
-rw------- rcapra/users    115 2007-01-17 16:57:19 ./rcapra-p2-styles.css
```

When you are satisfied that your tgz file is okay, you can submit it electronically through Sakai by going to the Assignments area and finding the "P2" assignment. After you think you have submitted the assignment, I strongly recommend checking to be sure the file was uploaded correctly by downloading it from Sakai and verifying it again with the "tar tvzf" command. Keep in mind that if I cannot access your files, I cannot grade them.

If for some reason you need to re-submit your homework file, you must add a version number to your filename. Sakai is configured so that it will only accept 3 total submissions. Use the following file naming convention if you need to re-submit:

Your first submission:	`youronyen-p2.tgz`
Your second submission:	`youronyen-p2-v2.tgz`
Your third submission:	`youronyen-p2-v3.tgz`

Sakai is also configured with a due date and an "accept until" date. Submissions received after the due date (even just 1 minute!) will receive a 10% penalty per day. The "accept until" date is 5 days after the due date. Submissions will not be accepted after the "accept until" date and will have a score of zero recorded.

If for some reason you are unable to submit an assignment to Sakai, as a last resort you may email it to the instructor along with a note about the problem you encountered. Then, as soon as you are able to, it is your responsibility to submit the exact same assignment to Sakai. The email serves as a record that you tried to submit the assignment on time, but to receive credit, your assignment must be uploaded to Sakai.