

NOTICE TO PODCAST SUBSCRIBERS

The source code for CS 50's library as well as for Fall 2007's problem sets in general can be found at <http://cs50.tv/>.

Problem Set 7: C\$50 Finance

out of 76 points

due by 7:00 P.M. on Friday, 7 December 2007

This problem set has more pages than past problem sets, but most contain narrative.
This is your last problem set ever for CS 50. :- (

Goals.

The goals of this problem set are to:

- Get you on that Interweb.¹
- Introduce you to XHTML, CSS, PHP, and SQL.
- Teach you how to teach yourself other languages.
- Play THE BIGGER BOARD.

Recommended Reading.

Per the syllabus, no books are required for this course. If you feel that you would benefit from some supplementary reading, though, below are some recommendations.

- <http://www.w3schools.com/xhtml/>
- <http://www.w3schools.com/css/>
- <http://www.w3schools.com/php/>
- <http://www.w3schools.com/sql/>

Collaboration.

For this problem set and this problem set only, you are welcome and encouraged to consult “outside resources,” including books, the Web, strangers, and friends, as you teach yourself more about XHTML, CSS, PHP, and SQL, so long as your work overall is ultimately your own. In other words, there remains a line, even if not precisely defined, between learning from others and presenting the work of others as your own.

You may adopt or adapt snippets of code written by others (whether found in some book, online, or elsewhere), so long as you cite (in the form of XHTML, CSS, or PHP comments) the origin thereof.

And you may learn from your classmates, so long as moments of counsel do not devolve into “show me your code” or “write this for me.” If in doubt as to the appropriateness of some discussion, contact the staff.

¹ <http://en.wikipedia.org/wiki/Interweb>

Academic Honesty.

All work that you do toward fulfillment of this course's expectations must be your own unless collaboration is explicitly allowed (*e.g.*, by some problem set or the final project). Viewing or copying another individual's work (even if left by a printer, stored in an executable directory, or accidentally shared in the course's virtual terminal room) or lifting material from a book, magazine, website, or other source—even in part—and presenting it as your own constitutes academic dishonesty, as does showing or giving your work, even in part, to another student.

Similarly is dual submission academic dishonesty: you may not submit the same or similar work to this class that you have submitted or will submit to another. Moreover, submission of any work that you intend to use outside of the course (*e.g.*, for a job) must be approved by the staff.

All forms of cheating will be dealt with harshly.

You are welcome to discuss the course's material with others in order to better understand it. You may even discuss problem sets with classmates, but you may not share code. In other words, you may communicate with classmates in English, but you may not communicate in, say, C. If in doubt as to the appropriateness of some discussion, contact the staff.

Grading Metrics.

Each question is worth the number of points specified parenthetically in line with it.

Your responses to questions requiring exposition will be graded on the basis of their clarity and correctness. Your responses to questions requiring code will be graded along the following axes.

Correctness. To what extent does your code adhere to the problem's specifications?

Design. To what extent is your code written clearly, efficiently, elegantly, and/or logically?

Style. To what extent is your code commented and indented, your variables aptly named, *etc.*?²

Rest assured that grades will be normalized across sections at term's end.

² Because lines of XHTML and PHP tend to be long, it's fine for this problem set if your lines of code exceed 80 characters.

Getting Started.

0. (1 point.) For a free point, do not read this sentence!³
1. By default, `www.people.fas.harvard.edu` isn't configured to support PHP 5, the latest version of PHP. And so CS 50 has installed its own copy of PHP 5's interpreter for you to use, much like the course installed its own copy of GCC and other tools for you earlier in the semester. To add support for PHP to your FAS account, SSH to `nice.fas.harvard.edu` and execute the command below.

```
cs50webify
```

You will be prompted for a password. Choose something that you will remember!⁴ Because work that you do for this problem set will, by nature, be accessible on the World Wide Web, this password will allow you to keep some of that work somewhat private.

Not only did that command create a directory called `public_html` in your home directory (if you didn't already have one), it also created a directory called `cs50` within that `public_html` directory.

Next execute the command below to copy this problem set's framework into `~/public_html/cs50/`.

```
cp -r ~/cs50/pub/ps/distributions/ps7 ~/public_html/cs50
```

Navigate your way to `~/public_html/cs50/ps7/` by executing the command below.

```
cd ~/public_html/cs50/ps7/
```

Then run `ls`. You should see the below. If you don't, don't hesitate to ask the staff for assistance.

```
css/  images/  inc/  index.php*  login2.php*  login.php*  logout.php*
```

Note that you have not (and will not) create a directory called `ps7` in `~/cs50/` for this problem set. Rather, you have created a directory called `ps7` in `~/public_html/cs50/`. All of the work that you do for this problem set will reside in (and be submitted from) `~/public_html/` and `~/public_html/cs50/ps7/`.

Now execute the command below.

```
cd inc/
```

³ D'oh! Okay, to get your free point, read this footnote.

⁴ If you do forget that password at any point, simply re-run `cs50webify`, and you'll be prompted to choose a new password (without affecting your files).

Then run `ls`. You should see the below.

```
apology.inc  common.inc  constants.inc  helpers.inc  stock.inc
```

Open up `constants.inc` with Nano (or your favorite text editor) and notice that the values of three variables (`DB`, `DBUSER`, and `DBPASS`) are currently missing. Head to the URL below.

```
http://www.fas.harvard.edu/~cs50/mysql/
```

You may be prompted to authenticate with your HUID and PIN. Ultimately, you'll reach a page that provides you with personalized values for each of those variables (two of which happen to be your FAS username). Fill in those blanks in `constants.inc` by copying and pasting these values, taking care to keep the variables' values enclosed in quotes. Then save your changes and quit Nano. You just configured your copy of C\$50 Finance to use your very own database! We'll explain the rest of that file in a bit.

Now run the command below to ensure that everything you just created and copied is indeed accessible on the World Wide Web.

```
fixwebfiles
```

Finally, surf on over to the URL below, where username is your FAS username.

```
http://www.people.fas.harvard.edu/~username/cs50/ps7/
```

Note that, thanks to `cs50webify`, the above URL is actually equivalent to the below (but shorter to type).

```
http://www.people.fas.harvard.edu/~username/cs50/ps7/index.php
```

When prompted, enter your FAS username and the password that you chose for `cs50webify`. You'll be "automagically" redirected to

```
http://www.people.fas.harvard.edu/~username/cs50/ps7/login.php
```

where you should be greeted with the login page for your copy of C\$50 Finance! If you don't, don't hesitate to ask the staff for assistance. Don't try to register, though, or log in just yet.

2. (5 points.) Surf on over to the URL below.

```
http://www.computersciencel.org/
```

You'll find yourself on the website for Computer Science E-1: Understanding Computers and the Internet, a course I used to teach for Harvard Extension School. Per that course's syllabus, "This course is all about understanding: understanding what's going on inside your computer when you flip on the switch, why tech support has you constantly rebooting your computer, how everything you do on the Internet can be watched by others, and how your

computer can become infected with a worm just by turning it on. In this course we demystify computers and the Internet, along with their jargon, so that students understand not only what they can do with each but also how it all works and why. Students leave this course armed with a new vocabulary and equipped for further exploration of computers and the Internet.”

The site’s left-hand column contains links to the course’s lectures on hardware, the Internet, multimedia, security, website development, and programming. (Okay, and Jeopardy and Pictionary.) The site’s right-hand column contains links to the course’s “videos of the week,” bite-sized segments on topics related to lectures.

Pick some lecture or video of the week whose topic might interest you and watch a few minutes in Flash or QuickTime format. Then, send a short email to your teaching fellow, CCing `cs50+ps7@fas.harvard.edu`, “teaching” her or him what you just learned (or, at least, watched). In just a few sentences, convince us that you do (now) know what you’re talking about!

3. (10 points.) Anytime between 11:00 A.M. on Monday, 3 December 2007, and this problem set’s deadline, surf on over to the URL below.

<http://www.fas.harvard.edu/~cs50/surveys/ps7/>

Rest assured that this survey is way, way shorter than Problem Set 5’s. We’d just like to tie up some loose ends. Please do take a few minutes to provide candid answers to the survey’s questions. Although you will be prompted to authenticate with your HUID and PIN, your answers, immediately upon submission, will be anonymized. We will know that you took the survey, but we will not know which answers are yours.

Home, sweet home.

4. (5 points.) It’s time to make yourself a home page!⁵ Navigate your way to `~/public_html/`. If you list the contents of your current working directory, you should see only that `cs50` directory that `cs50webify` created for you (unless you yourself created `~/public_html/` in the past and put something in it).

Create a file called `index.html` in that directory using Nano and fill that file with valid XHTML! In other words, whip yourself up a home page. Works of art, though encouraged, are by no means required. I didn’t exactly set the artistic bar very high in lecture, after all. So long as your XHTML is valid, your home page may contain as much or as little actual content as you would like.

When ready to examine your masterpiece (or work in progress), save your file and quit Nano. Before doing anything else, though, execute the command below.

```
ls -l
```

⁵ If you already have a home page on `www.people.fas.harvard.edu`, you need not create a new one for this problem, even if what exists is not valid XHTML. But do let your teaching fellow know, so that he or she knows what to expect.

The output you see should resemble the below.

```
drwx--x--x 2 jharvard student 4096 1636-11-30 19:01 cs50/  
-rw----- 1 jharvard student 247 1636-11-30 19:02 index.html
```

In the past, you’ve probably ignored the sequence of ten symbols (mostly hyphens) prefixing each line of `ls`’s long output. No longer! It turns out that each sequence represents a set of “permissions” that govern who (besides you) can read (r), write (w), or execute (x) access some file or directory.^{6,7} Linux lets you specify separate permissions for a file’s or directory’s owner (*i.e.*, you), for a file’s or directory’s group (*e.g.*, you plus all other students), and for the “world” (*i.e.*, you plus anyone with access to `nice.fas.harvard.edu` or to, in the case of files and directories inside `~/public_html/`, the World Wide Web).

By default, files that you create with Nano are readable and writable only by you, their owner. In fact, take a look at `index.html`’s current set of permissions: `-rw-----`. This sequence confirms that `index.html` is indeed readable and writable (but not executable) by you, as the second symbol through fourth represent owner permissions. That the rest of those symbols are hyphens means that neither your group nor the world have any permissions at all, as the fifth symbol through seventh represent group permissions, and the eighth symbol through tenth represent world permissions.

Well that’s no good! If `index.html` belongs on the Web, you’d best give everyone permission to read it! Go ahead and execute yet again the below.

```
fixwebfiles
```

Now go ahead and execute yet again the below.

```
ls -l
```

The output you see should now resemble the below.

```
drwx--x--x 2 jharvard student 4096 1636-11-30 19:01 cs50/  
-rw-r--r-- 1 jharvard student 247 1636-11-30 19:02 index.html
```

The difference, of course, is that both your group and world can now read (but not write or execute) your home page! Confirm as much by (finally) surfing over to the URL below, where username is, again, your FAS username.

```
http://www.people.fas.harvard.edu/~username/
```

Wow, that page is ugly. (Okay, maybe it’s not.) But the point is that you are now on the Web! Make any improvements you’d like to `index.html`. You may certainly, but need not, employ

⁶ To read a file means to, well, read its contents; to read a directory means to list its contents. To write to a file means to change its contents; to write to a directory means to add another file or directory to it. To execute a file means to run it like a program; to execute a directory means to enter it, as with `cd`.

⁷ The first symbol in a sequence indicates whether the permissions describe a directory (d).

CSS. Anytime you save changes with Nano, simply reload your page in your browser. You should not need to run `fixwebfiles` again for this particular file.

Ultimately, just be sure that your page is (tentatively) valid XHTML. You will likely find the W3C's Markup Validation Service of assistance, the URL of which is below.

<http://validator.w3.org/>

Note that you won't be able to pass URLs like

<http://www.people.fas.harvard.edu/~username/cs50/ps7/>

to that service, as pages therein will be password-protected.

Oh, by the way, it's no fun to be on the Web if nobody knows it. Go update your Facebook profile or at least email someone the URL of your new home!

5. It turns out that `fixwebfiles` is not a standard command but, rather, a "shell script" (*i.e.*, an interpreted program) that we wrote and installed on `nice.fas.harvard.edu` so that you can change the permissions of absolutely everything in `~/public_html/` with just a single command.⁸ This script of ours simply executes a standard command called `chmod` to change files' and directories' modes (*i.e.*, permissions). In particular, our script executes

```
chmod 644 foo
```

for every non-PHP file, `foo`, within `~/public_html/`,

```
chmod 700 bar
```

for every PHP file, `bar`, within `~/public_html/` and

```
chmod 711 baz
```

for every directory, `baz`, within `~/public_html/`.

What's with all those numbers? Well, 644 happens to mean `rw-r--r--`, and so all XHTML files are made readable and writable by you and just readable by everyone else; 711 happens to mean `rw-x--x--x`, and so all directories are made readable, writable, and executable by you and just executable by everyone else; and 700 happens to mean `rw-x-----`, and so all PHP files are made readable, writable, and executable only by you. Wait a minute, don't we want everyone to be able to read (*i.e.*, interpret) and execute your PHP files? Nope! For security reasons, PHP-based webpages are executed "as you" (*i.e.*, under your username) on `www.people.fas.harvard.edu`, no matter who pulls them up in a browser.

⁸ FAS IT actually has their own version of a script by this name on `nice.fas.harvard.edu`, but we have configured your FAS account to use our version instead, as ours supports PHP.

Okay, still, what's with all those numbers? Well, think of `rw-r--r--` as representing three triples of bits, the first triple of which, to be clear, is `rw-`. Imagine that `-` represents 0, whereas `r`, `w`, and `x` represent 1. And, so, this same triple (`rw-`) is just 110 in binary, or 6 in decimal! The other two triples, `r--` and `r--`, then, are just 100 and 100 in binary, or 4 and 4 in decimal! How, then, to express a pattern like `rw-r--r--` with numbers? Why, with 644!

Actually, this is a bit of a white lie. Because you can represent only eight possible values with three bits, these numbers (6, 4, and 4) are not actually decimal digits but “octal.” So you can now tell your friends that you speak not only binary, decimal, and hexadecimal, but octal as well.

Henceforth, you are welcome to continue using `fixwebfiles` if you'd rather not speak octal. But know that, when you leave Harvard, you leave `fixwebfiles` behind!

In other words, anytime you put a new file in `~/public_html/` or some subdirectory thereof (as with Nano or SFTP), be sure to set its permissions with `fixwebfiles` or `chmod`!

Show me the money!

6. Do not forget that this course has an anonymized bulletin board! Particularly for this problem set, turn to it for counsel and hints!
7. If you're not quite sure what it means to buy and sell stocks (*i.e.*, shares of a company), surf on over to the URL below for a tutorial!

<http://www.investopedia.com/university/stocks/>

You're about to implement C\$50 Finance, a Web-based tool with which you can manage portfolios of stocks! Not only will this tool allow you to check real stocks' actual prices and portfolios' values, it will also let you buy (okay, “buy”) and sell (fine, “sell”) stocks!⁹

8. Our weekly gift to you!
Big Press Release just out.

```
Inf inex Vent ures Inc. (INFX.OB)
Price:  0.83
5 day expected price 1.90
This one will run for sure
```

```
Starting a new Marketing Campaign.
The World will know about this company
Should you wait until to late?
```

This one did very well during last marketing campaign. Very Well!

With this News we expect prices to exceed our projected price of 1.90

⁹ Actually, quotes will be slightly delayed since we won't be paying for, say, Bloomberg.

9. Talk about a hot stock tip, let's get in on this opportunity now. Head on over to Yahoo! Finance at the URL below.

<http://finance.yahoo.com/>

Type the symbol for Infinex Ventures Inc., **INFX.OB**, into the text field in that page's top-left corner and click **GET QUOTES**. Odds are you'll see a table like the below.



Needless to say, I lost a lot of money because of that spam. Anyhow, notice how Yahoo reports not only a stock's most recent price (**Last Trade**) but also the time at which the stock last changed hands (**Trade Time**), the percent by which the stock's price changed over the course of the most recent (business) day (**Change**), the most recent (business) day's opening price (**Open**), the most recent (business) day's high and low prices (**Day's Range**), and more. Notice, too, that Yahoo lets you download this data. Go ahead and click **Download Data** to download a file in CSV format (*i.e.*, as comma-separated values). Open the file in Excel or any text editor (*e.g.*, Notepad or TextEdit), and you should see a "row" of values, all excerpted from that table. It turns out that the link you just clicked led to the URL below.

<http://download.finance.yahoo.com/d/quotes.csv?s=INFX.OB&f=s1l1d1t1c1ohgv&e=.csv>

Notice how Infinex's symbol is embedded in this URL (as the value of the HTTP parameter called *s*); that's how Yahoo knows whose data to return. Notice also the value of the HTTP parameter called *f*; it's a bit cryptic (and apparently undocumented), but the value of that parameter tells Yahoo which fields of data to return to you.

Here, we think, are the fields that can appear in that otherwise cryptic string, along descriptions thereof¹⁰

a	Ask
a2	Average Daily Volume
a5	Ask Size

¹⁰ Thanks to Chris Thorpe of SEAS for this list. Do let me know if you find this list incomplete or inaccurate.

Computer Science 50: Introduction to Computer Science I
Harvard College
Fall 2007

b	Bid
b4	Book Value
b6	Bid Size
c	Change & Percent
c1	Change
c3	Commission
c5	Pre Market Change and Percent
c8	After Hours Change and Percent
d	Dividend/Share
d1	Last Trade Date
e	Earnings/Share
e7	EPS Est. Current Yr
e8	EPS Est. Next Year
e9	EPS Est. Next Quarter
f6	Float Shares
g	Day's Low
g1	Holdings Gain & Percent
g3	Annualized Gain
g4	Holdings Gain
h	Day's High
i	More Info
j1	Market Capitalization
j4	EBITDA
j5	Change from 52-Week Low
j6	Percent Change from 52-Week Low
k3	Last Trade Size
k4	Change From 52-wk High
k5	Percent Chg From 52-wk High
l	Last Trade (With Time)
l1	Last Trade (Price Only)
l2	High Limit
l3	Low Limit
l8	Last Trade with Time (Pre/After Market)
m	Day's Range
m3	50-day Moving Average
m4	200-day Moving Average
m5	Change From 200-day Moving Average
m6	Percent Change From 200-day Moving Average
m7	Change From 50-day Moving Average
m8	Percent Change From 50-day Moving Average
n	Name
n4	Notes
o	Open
p	Previous Close
p1	Price Paid
p5	Price/Sales
p6	Price/Book
q	Ex-Dividend Date
r	P/E Ratio
r1	Dividend Pay Date
r5	PEG Ratio
r6	Price/EPS Est. Current Year
r7	Price/EPS Est. Next Year
s	Symbol
s1	Symbol

s1	Shares Owned
s7	Short Ratio
t1	Last Trade Time
t6	Trade Links
t7	Ticker Trend
t8	1-Year Target Price
v	Volume
v1	Holdings Value
w	52-Week Range
w1	Day's Value Change
y	Dividend Yield

It's worth noting that a lot of websites that integrate data from other websites do so via "screen scraping," a process that requires writing programs that parse (or, really, search) HTML or XHTML for data of interest (*e.g.*, air fares, stock prices, *etc.*). Writing a screen scraper for a site tends to be a nightmare, though, because a site's markup is often a mess, and if the site changes the format of its pages overnight, you need to re-write your scraper.

Thankfully, because Yahoo provides data in CSV, C\$50 Finance will avoid screen scraping altogether by downloading (effectively pretending to be a browser) and parsing CSV files instead. Even more thankfully, we've written that code for you!

In fact, let's turn our attention to the code you've been given.

10. Navigate your way to `~/public_html/cs50/ps7/` and open up `index.php` with Nano. You'll see XHTML for a pretty simple page, the same page you tried to pull up earlier when testing your framework (just before you were redirected to `login.php`). Notice the references to `styles.css` and `logo.gif`. Those files can be found in `~/public_html/cs50/ps7/css/` and `~/public_html/cs50/ps7/images/`, respectively. We placed those two files into subdirectories in the interests of keeping `~/public_html/cs50/ps7/` tidy. As you proceed to implement C\$50 Finance, you're welcome to drop additional files into either directory.

Notice also that `index.php`'s very first line is (necessarily) the below, a "shebang."

```
#!/home/c/s/cs50/pub/local/alpha/php/bin/php-cgi
```

Any PHP that you create for this problem set whose name ends in `.php` (but not `.inc`) must begin with this line.

Recall that this line tells `www.people.fas.harvard.edu` to use the course's installation of PHP in order to execute the file. Moreover, it tells the server to use the installation of PHP that we compiled specifically for DEC Alpha CPUs, which is what's inside `www.people.fas.harvard.edu`. Because `www.courses.fas.harvard.edu` has Intel CPUs, the course's own website actually uses a different installation of PHP, compiled specifically for x86.

Notice finally that `index.php` “requires” (*i.e.*, includes) a file called `common.inc`, which can be found in `~/public_html/cs50/ps7/inc/`. Any file that you create for this problem set whose name ends in `.php` must also contain, before any other code, that very same line, excerpted below.

```
require_once("inc/common.inc");
```

Note that if you decide to place PHP files within subdirectories of `~/public_html/cs50/ps7/`, you may need to specify a different path for `common.inc` (*e.g.*, `../inc/common.inc`).

Let’s take a look at the code we’re requiring via `require_once`. Navigate your way to `~/public_html/cs50/ps7/inc/` and open up `common.inc` with Nano. Because `index.php` requires this file (via that call to `require_once`), every one of this file’s lines will be executed before anything else in `index.php`. Notice first that `common.inc` does not begin with a shebang. Because this file is not accessed (and thus executed) directly via browsers, it is neither necessary nor correct to include the shebang. This file’s first few lines of actual code ensure that you’ll be informed of errors in your own code via your browser. The next few lines require yet three other files; we’ll return to those shortly. The call to `session_start` ensures that you’ll have access to `$_SESSION`, a “superglobal” variable via which we’ll remember that a user is logged in.¹¹ The next lines of code ensure that users will be required to log in. The last lines of code connect to your database, where you’ll store users’ portfolios.

Alright, now open `constants.inc` with Nano. In this file have we defined some global constants, three of whose values you yourself provided earlier. Because all of your PHPs shall require `common.inc`, which, in turn, requires `constants.inc`, you will have access to this file’s globals from each of your PHPs. Notice the URLs for MSN and Yahoo, the latter of which should look awfully familiar. Missing from `YAHOO`, though, is a value for that parameter `s`. Let’s see why that is.

Open up `stock.inc` with Nano, and you’ll see something that resembles a C struct. Indeed, this code defines a structure (a “class” in PHP) for stocks. Although Yahoo provides many more fields than those encapsulated in this structure, our framework, out of the box, provides only the basics.

Now take a look at `helpers.inc` with Nano. You need not understand how all of that code works, but make sure you understand what its functions can do for you by reading, at least, comments therein. Notice, in particular, how `lookup` expects, as its sole argument, a stock’s symbol, which it appends to `YAHOO` using PHP’s concatenation operator (`.`) in order to download the right CSV!

¹¹ Even though HTTP is a “stateless” protocol, whereby browsers are supposed to disconnect from servers as soon as they’re done downloading pages, “cookies” allow browsers to remind servers who they (or, really, you) are on subsequent requests for content. PHP uses “session cookies” to provide you with `$_SESSION`, an associative array in which you can store any data to which you’d like to have access for the duration of some user’s visit. The moment, say, Lord Dark Helmet ends his “session” (*i.e.*, visit) by closing his browser, the contents of `$_SESSION` are lost for Lord Dark Helmet specifically because the next time he visits, he’ll be assigned a new cookie!

Finally, take a peek at `apology.inc`. This file serves as a “template” for `apologize` in `helpers.inc` so that, via just one function, you can apologize to users for (*i.e.*, report) all sorts of problems.

Alright, only three files remain to examine, each of which ends in `.php`! Navigate back to `~/public_html/cs50/ps7/`, and open up `login.php` with Nano. Recall that you were redirected to this page when you tried to pull up `index.php` with your browser. Notice that this file doesn’t contain too much code. In fact, much like `index.php`, it’s almost entirely XHTML. But it’s that XHTML that implements that login page that you saw. Note that it lays out a form using an “invisible” table. In fact, if you’d like, change

```
border="0"
```

to

```
border="1"
```

in the start tag for that table. Save the file, then revisit the URL below, reloading if necessary.

<http://www.people.fas.harvard.edu/~username/cs50/ps7/login.php>

The table should now be visible. Notice next the far more important line excerpted below.

```
<form action="login2.php" method="post">
```

This line instructs your browser to “submit” the form’s data to `login2.php` via POST. It must be the case, then, that `login2.php` handles authentication of users. Let’s check. Open up `login2.php` with Nano.

It turns out that `login2.php` isn’t terribly long. Its first line of code, just like `index.php` and `login.php`, requires that file called `common.inc`. Its next lines of code “escape” the user’s input for safety using `mysql_real_escape_string`, lest C\$50 Finance’s database fall victim to a “SQL injection attack,” whereby a user submits SQL instead of a username and/or password. See http://www.php.net/mysql_real_escape_string for reference.

The next line of code prepares a string of SQL as follows.

```
$sql = sprintf("SELECT uid FROM users WHERE username='%s' AND password='%s'",  
               $username, $password);
```

To be clear, suppose that Lord Dark Helmet tries to log into C\$50 Finance. That call to `sprintf` will create and return the following string.

```
SELECT uid FROM users WHERE username='dhelmet' AND password='12345'
```

See <http://www.php.net/sprintf> for reference.

Perhaps needless to say, `login2.php`'s next line of code executes that `SELECT` with `mysql_query`, storing the “result set” (*i.e.*, rows returned) in a variable called `$result`. Only if Lord Dark Helmet's username is `dhelmet` and password is `12345`, though, should the database return an actual row. And, so, if `mysql_num_rows` returns 1, Lord Dark Helmet has successfully authenticated! Our code “remembers” as much by storing his numeric user ID (`uid`) in `$_SESSION`; it then redirects him to `index.php`, where his portfolio (once you implement it) awaits!¹² If, however, his password (or perhaps username) was invalid, he is instead redirected to `login.php?username=dhelmet`, where he is prompted to try again. By redirecting him back to `login.php` with this HTTP parameter called `username`, we empower `login.php` to pre-populate part of its form so that Lord Dark Helmet need not type his username again. Of course, if he mistyped his username, he'll have to correct it himself.

Incidentally, why does this redirection back to `index.php`, upon successful authentication, not result in an infinite loop? Well, recall that `index.php` requires `common.inc`, which contains the following code.

```
if (!preg_match("/(?:log(?:in|out)|register)\d*\..php$/", $_SERVER["PHP_SELF"]))
{
    if (!isset($_SESSION["uid"]))
        redirect("login.php");
}
```

Though a bit scary (I prefer “elegant”), that code simply asks whether `$_SESSION["uid"]` has been assigned any value (*e.g.*, Lord Dark Helmet's user ID). If not, it must be that no one's logged in, else `login2.php` would have assigned it a value, and so we had best redirect traffic to `login.php` (by calling `redirect`, a function defined in `helpers.inc`). If, though, `$_SESSION["uid"]` is indeed set, we won't redirect but will, instead, leave the logged-in user wherever he or she is. Of course, if the user is already at `login.php`, `logout.php`, or `register.php`, this code won't redirect either.

Now, how do we enable Lord Dark Helmet to log out? Why, with `logout.php`! Take a look at that file with Nano. Note that all we need do is invoke `session_destroy`, the function of which is to obliterate (*i.e.*, erase the contents of) `$_SESSION`. Alternatively, Lord Dark Helmet can simply close his own browser, as `$_SESSION` is destroyed in that case as well.

11. Phew, that was a lot. Time for a snack?
12. Alright, let's talk about that database we keep mentioning. So that you have someplace to store users' portfolios, we've taken the liberty of creating a MySQL database just for you for this problem set. We've even pre-populated it with one table!

Surf on over to the URL below, again authenticating with your HUID and PIN if prompted.

<http://www.fas.harvard.edu/~cs50/mysql/>

¹² Rather than “remember” users by way of their usernames (which are, by nature, strings), you'll see that we instead rely, for efficiency's sake, on “user IDs” (which are integers) that uniquely identify users.

Ultimately, you'll reach a page that reminds you of your database's name as well as the username and password via which you can connect to your database. So that it is unique among all databases, your database's name is, again, simply your username.

Follow the link to **phpMyAdmin**, a Web-based tool (that happens to be written in PHP) with which you can manage your MySQL database. Type or paste in your database's username and password, then click **Go!** You'll ultimately find yourself at phpMyAdmin's main page.

Notice, at top-left, a link to your database (*i.e.*, username), beneath which is a link to a table (that we created for you) called **users**. Click the latter to examine that table's structure. Notice anything familiar? You should! Recall `login2.php` generates, via `sprintf`, queries like the below.

```
SELECT uid FROM users WHERE username='dhelmet' AND password='12345'
```

As phpMyAdmin makes clear, this table called **users** contains three fields: `uid` (the type of which is an unsigned `INT`) along with `username` and `password` (both of whose types is `VARCHAR`). It appears that none of these fields is allowed to be `NULL`, and the maximum length for each of each of `username` and `password` is 255. A neat feature of `uid`, meanwhile, is that it is auto-incrementing: when inserting a new user into the table, you needn't specify a value for `uid`; the user will be assigned the next available `INT`. Notice, finally, the box labeled **Indexes**. It appears that this table's "primary key" is `uid`, the implication of which is that (as expected) no two users can share the same user ID.¹³ Of course, `username` should also be unique across users, and so we have also defined it as just that. To be sure, we could have defined `username` as this table's primary key. But, for efficiency's sake, the more conventional approach is to use an `INT` like `uid`. Incidentally, these fields are called "indexes" because, for primary keys and otherwise unique fields, databases tend to build "indexes," data structures that enable them to find rows quickly by way of those fields.

Make sense? Okay, let's see if any users exist! Click the tab labeled **Browse** to see the contents of this table. Ah, some familiar folks. In fact, there's Lord Dark Helmet's username and password! Head on back to











`http://www.people.fas.harvard.edu/~username/cs50/ps7/login.php`

and try to log in as Lord Dark Helmet. If successful, you'll find yourself at `index.php`, where (for the moment) very little awaits.

13. Head on back to phpMyAdmin and click the tab labeled **Structure** for that table called **users**. Let's give each of your users some cash. In that page's middle is a form with which you can **Add ... field(s) ... After** another. Click the radio button beside **After**, select `password` from the drop-down menu, then click **Go**.

¹³ A primary key is a field with no duplicates (*i.e.*, that is guaranteed to identify rows uniquely).

Via the form that appears, add a field of type `DECIMAL` called `cash`, using the settings depicted below.

Field	cash
Type 	DECIMAL 
Length/Values ¹	65, 2
Collation	
Attributes	UNSIGNED 
Null	not null 
Default ²	0.00
Extra	
	<input type="radio"/>
	<input type="radio"/>
	<input type="radio"/>
---	<input checked="" type="radio"/>
	<input type="checkbox"/>
Comments	

It probably makes sense intuitively that a field called cash should be of type DECIMAL and that its values should be both unsigned and non-NULL. (See <http://dev.mysql.com/doc/refman/5.0/en/numeric-types.html> for reference.) Our specification of 65 and 2 limits everyone's cash to just, heh,
\$999,999.
By default, though, everyone will have \$0.00. Click **Save** to make all this so. Then return to the tab labeled **Browse** and give everyone \$10,000.00. The easiest way is to click **Check All**, then click the icon on that same line that looks like pencil. On the page that appears, change 0.00 to 10000.00 for each of your users, then click **Go**. Won't they be happy!

14. (10 points.) It's now time to code! Let's empower new users to register.

Return your attention to `nice.fas.harvard.edu` and navigate your way to `~/public/html/cs50/ps7/`. Execute

```
cp login.php register.php
```

followed by

```
cp login2.php register2.php
```

to jumpstart this process. Then run `fixwebfiles` (or `chmod`) to ensure all permissions are set properly. Open up `register.php` with Nano and change the head's title as you see fit. Then change the value of the form's action attribute from `login2.php` to `register2.php`. Next add an additional row to the XHTML table containing a new field called `password2`.

(After all, we probably want registrants to type passwords twice to discourage mistakes.) Finally, change the submit button's value from Log In to Register and make that bottommost anchor (*i.e.*, link) point back to login.php (so that users can navigate away from this page if they already have accounts).

Alright, let's take a look at your work! Bring up

<http://www.people.fas.harvard.edu/~username/cs50/ps7/login.php>

and click that page's link to register.php. If that page appears broken (or perhaps simply ugly), feel free to make further tweaks using Nano, saving your changes, thereafter reloading the page.

Once the page looks okay, head back to Nano and open up register2.php. Needless to say, you need to replace the code there so that it actually registers users. Allow us to offer some hints.

- If `$_POST["username"]` or `$_POST["password"]` is blank or if `$_POST["password"]` does not equal `$_POST["password2"]`, you'll want to return to the registrant a page that apologizes, explaining at least one of the problems.
- To insert a new user into your database, you might want to pass `sprintf` a string like `INSERT INTO users (username, password, cash) VALUES('%s', '%s', 10000.00)`, though we leave it to you to decide how much cash to give new users.
- Know that `mysql_query` will return `FALSE` if your `INSERT` fails (as can happen if, say, username already exists).¹⁴ Of course, if you cannot `INSERT`, you should certainly apologize.
- If, though, your `INSERT` succeeds, know that you can find out which `uid` was assigned to that user with a call to `mysql_insert_id` right after your call to `mysql_query`.¹⁵
- If registration succeeds, you might as well log the new user in (as by “remembering” that `uid` in `$_SESSION`), thereafter redirecting to `index.php`.

All done with the above? Ready to test? Head back to

<http://www.people.fas.harvard.edu/~username/cs50/ps7/register.php>

and try to register a new username. If you reach `index.php`, odds are you done good! Confirm as much by returning to phpMyAdmin, clicking once more that tab labeled **Browse** for the table called `users`. May that you see your new user! If not, it's time to debug. ; -)

¹⁴ See http://www.php.net/mysql_query for reference.

¹⁵ See http://www.php.net/mysql_insert_id for reference.

Recall, incidentally, that `helpers.inc` provides a function called `dump` that spits out to your browser the value(s) in any variable you pass it. For instance, if you'd like to dump the entire contents of `$_POST`, simply include

```
dump($_POST);
```

wherever you'd like. Note that this function is meant for debugging, not apologies to users!

15. Do bear in mind as you proceed further that you are welcome to play with and learn from the staff's implementation of C\$50 Finance, available at the URL below.

<http://www.fas.harvard.edu/~cs50/finance/>

In particular, you are welcome to register with as many (fake) usernames as you would like in order to play. And you are welcome to view our pages' XHTML and CSS (by viewing our source using your browser) so that you might learn from or improve upon our own design. If you wish, feel free to adopt our XHTML and CSS as your own.

But do not feel that you need copy our design. In fact, for this problem set, you may modify every one of the files we have given you to suit your own tastes as well as incorporate your own images and more. In fact, may that your version of C\$50 Finance be nicer than ours!

In fact, you're even welcome to rename your baby.

<http://googlefont.com/>

16. Do not forget that this course has a bulletin board!
17. (10 points.) Now let's empower users to get quotes for individual stocks. Let's have you again copy `login.php` and `login2.php`, this time naming the copies `quote.php` and `quote2.php`, respectively. Again ensure that the copies' permissions are set properly. (Remember how?) Modify `quote.php` so that it submits just one text field called `symbol` to `quote2.php`. Then modify `quote2.php` so that, upon receipt of that `symbol`, it informs the user of the current price of the stock described by that `symbol`.

How now? Well, recall that function called `lookup` in `helpers.inc`. Why not invoke it with code like the below?

```
$s = lookup($_POST["symbol"]);
```

Assuming `$_POST["symbol"]` is non-null and contains a symbol for an actual stock, `lookup` will return an object of type `stock`. (Recall that `stock` was defined in `stock.inc`.) If you think of `$s` as a pointer (a "reference" in PHP), you can access (or, better yet, print) individual fields in that object with code like the below.

```
print($s->price);
```

Incidentally, know that your PHP code need not appear only at the top of .php files. In fact, you can intersperse PHP and XHTML, as in the below, provided `$s` has already been assigned elsewhere (*e.g.*, atop your file) the return value of `lookup`.

```
<div align="center">
  A share of <? print($s->name); ?> currently costs $<? print($s->price); ?>.
</div>
```

Of course, if the user submits an invalid symbol (for which `lookup` returns `null`), be sure to apologize to the user, explaining the problem!

18. (5 points.) And now it's time to do a bit of design. At present, your database has no way of keeping track of users' portfolios, only users themselves.¹⁶ It doesn't really make sense to add additional fields to `users` itself in order to keep track of the stocks owned by users (using, say, one field per company owned). After all, how many different stocks might a user own? Better to maintain that data in a new table altogether so that we do not impose limits on users' portfolios or waste space with potentially unused fields.

Exactly what sort of information need we keep in this new table in order to “remember” users' portfolios? Well, we probably want a field for users' IDs (`uid`) so that we can cross-reference holdings with entries in `users`. We probably want to keep track of stocks owned by way of their symbols since those symbols are likely shorter (and thus more efficiently stored) than stocks' actual names.¹⁷ And we probably want to keep track of how many shares a user owns of a particular stock. In other words, a table with three fields (`uid`, `symbol`, and `shares`) sounds pretty good, but you're welcome to proceed with a design of your own. Whatever your decision, head back to `phpMyAdmin` and create this new table, naming it however you see fit. To create a new table, click your database's name (*i.e.*, your username) in `phpMyAdmin`'s top-left corner. Then, in the page's right-hand frame, specify the new table's **Name** and **Number of Fields**, then click **Go**. On the screen that appears, define (in any order) each of your fields.

If you decide to go with three fields (namely `uid`, `symbol`, and `shares`), realize that `uid` should not be defined as a primary key in this table, else each user could own no more than one company's stock (since his or her `uid` could not appear in more than one row). Realize, too, that you probably don't want some `uid` and some `symbol` to appear together in more than one row. Better to consolidate users' holdings by updating `shares` whenever some user sells or buys more shares of some stock he or she already owns. A neat way to impose this restriction on your table is to define a “joint primary key” by clicking the radio buttons next to the little icon with a key for both `uid` and `symbol`. That way, `mysql_query` will return `FALSE` if you try to insert more than one row for some pair of `uid` and `symbol`. We leave it to you, though, to decide your fields' types.¹⁸ When done defining your table, click **Save**!

¹⁶ By “portfolio,” we mean a collection of stocks (*i.e.*, shares of companies) that some user owns.

¹⁷ Of course, you could also assign unique numeric IDs to stocks and remember those instead of their symbols. But then you'd have to maintain your own database of companies, built up over time based on data from, say, Yahoo. It's probably best (and it's certainly simpler), then, to keep track of stocks simply by way of their symbols.

¹⁸ If you include `uid` in this table, know that its type should match that in `users`. But don't specify `auto_increment` for that field in this new table, as you only want auto-incrementation when user IDs are created (by `register2.php`) for new users.

19. (10 points.) Before we let users buy and sell stocks themselves, let's give some freebies to Lord Dark Helmet and friends. Click, in phpMyAdmin's left-hand frame, the link to `users`. Then click the tab labeled **Browse** and remind yourself (assuming your new table's design matches ours) of your current users' IDs. Then click, in phpMyAdmin's left-hand frame, the link to your new table, followed by the tab labeled **Insert**. Via this interface, go ahead and "buy" some shares of some stocks on behalf of your users by manually inserting rows into this table. (You may want to return to Yahoo! Finance to look up some actual symbols.) No need to debit their cash in `users`.

Once you've bought your users some shares, let's see what you did. Click the tab labeled **SQL** and run a query like the below, where `table` is your new table's name.¹⁹

```
SELECT * FROM table WHERE uid=1
```

Assuming 1 is still Lord Dark Helmet's user ID, that query should return all rows from `table` that represent the lord's holdings. If the only fields in `table` are, say, `uid`, `symbol`, and `shares`, then know that the above is actually equivalent to the below.

```
SELECT uid,symbol,shares FROM table WHERE uid=1
```

If, meanwhile, you'd like to retrieve only Lord Dark Helmet's shares of Infinex Ventures, you might like to try a query like the below.

```
SELECT shares FROM table WHERE uid=1 AND symbol='INFX.OB'
```

If you happened to buy the lord some shares of that company, the above should return one row with one column, the number of shares. If you did not think to get him in on that deal, the above will return an empty result set.

Incidentally, via this **SQL** tab, you could have inserted those "purchases" with `INSERT` statements. But phpMyAdmin's GUI saved you the trouble.

Alright, let's put this knowledge to use. It's time to let users peruse their portfolios! Overhaul `index.php`, in such a way that it reports each of the stock's in a user's portfolio, including number of shares and current value thereof, along with a user's current cash balance. You are welcome, though not required, to make use of the `stock` class's other data. Needless to say, `index.php` will need to invoke `lookup` much like `quote2.php` did, though perhaps multiple times. Know that a PHP can certainly invoke `mysql_query` multiple times, even though, thus far, we've seen it used in each file no more than once. Similarly can you call `mysql_fetch_array` multiple times, particularly in loops.

¹⁹ Incidentally, because 1 is a number (just as 10000.00 was earlier), you need not enclose it in quotes like you do strings.

For instance, if your goal is simply to display, say, Lord Dark Helmet's holdings, one per row in some XHTML table, you can generate rows with code like the below.²⁰

```
<?
    $result = mysql_query("SELECT symbol,shares FROM table WHERE uid=1");

    while ($row = mysql_fetch_array($result))
    {
        $s = lookup($row["symbol"]);
        print('<tr>');
        print('<td>' . $s->name . '</td>');
        print('<td>' . $row["shares"] . '</td>');
        print('</tr>');
    }
?>
```

Though commonly done, generating XHTML via calls to print isn't terribly elegant. An alternative approach, though still far from ideal, is code more like the below.

```
<? $result = mysql_query("SELECT symbol,shares FROM table WHERE uid=1"); ?>

<? while ($row = mysql_fetch_array($result)) { ?>

    <? $s = lookup($row["symbol"]); ?>

    <tr>
        <td><? print($s->name); ?></td>
        <td><? print($row["shares"]); ?></td>
    </tr>

<? } ?>
```

With that said, for this last problem set, focus more on correctness than on design or style.

Look, as before, to

<http://www.fas.harvard.edu/~cs50/finance/>

for inspiration or hints. But do not feel obliged to mimic our design. Make this website your own! Incidentally, though we keep using Lord Dark Helmet in examples, your code should work for whichever user is logged in!

20. (10 points.) And now it is time to implement the ability to sell in, say, `sell1.php` and `sell2.php`. We leave the design of the former, in particular, to you. But know, for the latter, that you can delete rows from your table (on behalf of, say, Lord Dark Helmet) with SQL like the below.

```
DELETE FROM table WHERE uid=1 AND symbol='INFX.OB'
```

²⁰ Note that developers tend to use single quotes around XHTML, lest the XHTML itself contain double quotes, as around attributes' values.

We leave it to you to infer exactly what that statement should do. Of course, you could try the above out via phpMyAdmin's **SQL** tab. Now what about the user's cash balance? Odds are, your user is going to want the proceeds of all sales. So selling a stock involves updating not only your table for users' portfolios but `users` as well. We leave it to you to determine how to compute how much cash a user is owed upon sale of some stock. But once you know that amount (say, \$500), SQL like the below should take care of the deposit (for, say, the lord).²¹

```
UPDATE users SET cash=cash+500 WHERE uid=1
```

It's fine, for simplicity, to require that users sell all shares of some stock or none, rather than only a few.

Needless to say, try out your code by logging in as some user and selling some stuff! You can always "buy" it back manually with phpMyAdmin!

21. (10 points.) And now for your big finale. Implement the ability to buy, in, say, `buy.php` and `buy2.php`.²² The interface with which you provide a user is entirely up to you, though, as before, feel free to look to

<http://www.fas.harvard.edu/~cs50/finance/>

for inspiration or hints. Of course, you'll need to ensure that a user cannot spend more cash than he or she has on hand. And you'll want to make sure that users can only buy whole shares of stocks, not fractions thereof. For this latter requirement, know that a call like

```
preg_match("/^\d+$/", $_POST["shares"])
```

will return `TRUE` if and only if `$_POST["shares"]` contains a non-negative integer, thanks to "regular expressions." See http://www.php.net/preg_match for reference. Take care, as always, to apologize to the user if you encounter some problem.

Incidentally, if you implemented your table for users' portfolios as we did ours (with that joint primary key), know that SQL like the below (which, unfortunately, wraps onto two lines) will insert a new row into `table` unless the specified pair of `uid` and `symbol` already exists in some row, in which case that row's number of shares will simply be increased (say, by 10).

```
INSERT INTO table (uid,symbol,shares) VALUES(1,'INFX.OB',10)
ON DUPLICATE KEY UPDATE shares=shares+VALUES(shares)
```

As before, be sure to bang on your code!

²¹ Of course, if the database or webserver happens to die between this `DELETE` and `UPDATE`, Lord Dark Helmet might lose out on all of that cash. You need not worry about such cases! It's also possible, because of multithreading and, thus, race conditions, that a clever Lord Dark Helmet could trick your site into paying out more than once. You need not worry about such cases either! Though, if you're so very inclined, you can employ InnoDB tables and SQL transactions. See <http://dev.mysql.com/books/mysqlpress/mysql-tutorial/ch10.html> for reference.

²² As before, you need not worry about interruptions of service or race conditions.

22. Do not forget that... Right, bulletin board.
23. Once you think you're all done, it's time to invite one or more friends to try out your site. Encourage them to try breaking it, like a good adversary would. Under no circumstances should they (or we) be able to crash your code (*i.e.*, trigger some warning or error from PHP's own interpreter). You'd best catch and/or apologize for any error that a user's input, malicious or otherwise, might induce!
24. If you're up for the challenge, why not bite off another feature or two? None of the below is required, and you'll be rewarded only with admiration and praise, but why not see if you can make your site even better? Odds are, you'll learn yet another new trick or two. Feel free to solicit counsel from the course's staff or bulletin board. In fact, you're welcome to collaborate outright with classmates on the features below, inasmuch as none is officially required of you. Interpret each feature below as you will!
- ☐ Empower users to change their passwords.
 - ☐ Empower users who've forgotten their password to receive reminders via email. See <http://phpmailer.codeworxtech.com/tutorial.html> for reference.
 - ☐ Email users "receipts" anytime they buy or sell stocks. See <http://phpmailer.codeworxtech.com/tutorial.html> for reference.
 - ☐ Empower users to deposit additional funds.
 - ☐ Keep users' passwords encrypted in your database. See <http://www.php.net/crypt>, <http://www.php.net/manual/en/ref.mcrypt.php>, and/or <http://dev.mysql.com/doc/refman/5.0/en/encryption-functions.html> for reference.
 - ☐ Present users with links to recent articles related to companies for which users have shares. You may find the stock class's items of interest.
 - ☐ Download yet additional fields from Yahoo, updating `helpers.inc` and `stock.inc` accordingly, and provide users with access to that data.
 - ☐ Integrate more sources of data than just MSN and Yahoo.
 - ☐ Implement the ability to short stocks!
 - ☐ ...
25. Just for fun, why don't we give you some cash out of our own pocket. How does \$10K for each of you sound? That's, what, only \$2,830,000? So we'll cut down on cupcakes and candy.

If you would like, surf on over to

<http://www.fas.harvard.edu/~cs50/finance/>

anytime after 11:00 A.M. on Monday, 3 December 2007, and you'll find that \$10K awaits you if you follow the link to **play THE BIGGER BOARD**. We shall see, come the course's final lecture on Friday, 14 December 2007, who exits this course with the most money in hand.

Submitting Your Work.

26. Ensure that your work is in `~/public_html/` and in `~/public_html/cs50/ps7/`.²³ Submit your work by executing the command below.

```
cs50submit ps7
```

Thereafter, follow any on-screen instructions until you receive visual confirmation of your work's successful submission. You will also receive a "receipt" via email to your FAS account, which you should retain until term's end. You may re-submit as many times as you'd like; each resubmission will overwrite any previous submission. But take care not to re-submit after the problem set's deadline, as only your latest submission's timestamp is retained.

Per #3, don't forget to complete the survey at the URL below!

`http://www.fas.harvard.edu/~cs50/surveys/ps7/`

²³ If you have files in `~/public_html/` other than those required for this problem set and you do not wish `cs50submit` to copy those files (particularly if private) into our account, simply remove them from `~/public_html/` temporarily before submitting your work.