Assignment 2: Better banking

Add some content to the web page you built for the previous assignment. The new page should have a form that invokes a CGI program called `bank-2.php`, written in PHP. Invoking the program as a simple URL should produce a page that looks like the screenshot at at `http://www.cs.uky.edu/~raphael/courses/CS316/project2/asg.bank-2.png`.

As you see, the bank's web page has gotten some new features and dropped some old ones. The bank uses a mySQL database with one relation called `accounts` that has two attributes: `checking` and `savings`, showing the current balance for each account. The mySQL data type for a field containing money is `decimal(15,2)`. The account has at present only one row, because there is only one bank client, who initially has $100 in the checking account and $1000 in the savings account.

The user has several options to add money and to move money between accounts. When the CGI program receives a request, it uses mySQL commands to accomplish the request and updates the table (bordered in 2px of solid blue).

Method

(1) As before, the HTML that your CGI program generates must be self-contained, without recourse to any external CSS stylesheets or JavaScript libraries. The web page must validate without warnings or errors.

(2) Your CGI program must be written in PHP. It must also be in a single file, except that it should have a line saying

```
require_once 'db_creds.inc';
```

The file `db_creds.inc` should have these lines:

```
<?php
define('K_USERNAME', 'abcd123');
define('K_PASSWORD', 'myPassword');
define('K_CONNECTION_STRING', 'mysql:host=mysql;port=3306;dbna
?>
```

where you should change `abcd123` in both places to your login name. The password should be 'u' followed by the last 7 digits of your UKID (like `u0123456`). You should turn in only your PHP program, not the `db_creds.inc` file.

(3) You already have a database in your *mysql* account with your name (like `abcd123`).

(4) Although PHP has global variables, you must import global variables into any function that needs them. For example:

```
function doSomething($param1, $param2) {
     global $pdo;
     ...
} // doSomething
```

(5) You must prevent cross-site scripting attacks. The best way is to use `?` in any mySQL statement that refers to data coming from the web page, then prepare the statement, then execute it providing the actual values, like this:

```
$sql = "UPDATE accounts SET checking = ?";
$prepared = $pdo->prepare($sql);
$prepared->execute([$valueFromHTMLform]);
```

(6) Your PHP program does not need to prevent accounts becoming negative.

(7) You do not need to use PHP classes; all the code may be in the main program, although your PHP program should be modular, with code nicely divided into functions.

(8) You can connect to your database directly using mySQL and executing commands (changing `abcd123` to your login name):

% mysql -h mysql.cs.uky.edu -D `abcd123` -u `abcd123` -p
[it will prompt for your password]
DESCRIBE accounts;
SELECT * FROM accounts;
QUIT;

(9) You may refer to online resources and books, but you must not share your work with others.

Extra

(10) Verify (in the PHP program) that numerical quantities are in fact numeric. If you don't verify data, then it is acceptable for your program to treat non-numeric values as 0.

(11) Prevent invalid requests, such as depositing a negative amount of money or allowing an account to have a negative balance.

(12) Add a `start fresh` button that resets the accounts to their initial values.

Turn in

Turn in your PHP program (a single file called `bank-2.php`) via Canvas.