Need data from the web - you want Perl

1. Write a Perl program which reads from STDIN a username then a password (not both simultaneously).

   It should then check the password matches one stored for user *username* in the file `users/` *username* `.password` .

   It should print a message indicating the user is unknown if this file does not exist otherwise print an appropiate message indicating whether the password matches.

   For example:

```
$ mkdir users
$ echo 'correct horse battery staple' >users/andrewt.password
$ login.pl
username: jas
password: beer
Unkown username!
$ login.pl
username: andrewt
password: 42
Incorrect password!
$ login.pl
username: andrewt
password: correct horse battery staple
You are authenticated.
```

   Sample Perl solution

```
#!/usr/bin/perl -w
print "username: ";
$username = <STDIN>;
chomp $username;
print "password: ";
$password = <STDIN>;
chomp $password;

# sanitize username
$username = substr $username, 0, 256;
$username =~ s/\W//g;
$password_file = "accounts/$username/password";
if (!open F, "<$password_file") {
    print "Unknown username!\n";
} else {
    $correct_password = <F>;
    chomp $correct_password;
    if ($password eq $correct_password) {
        print "You are authenticated.\n";
    } else {
        print "Incorrect password!\n";
    }
}
```

2. How are CGI scripts run?

   CGI script are run when a web server receives a request for a particular URL.
   Web servers can be configured to run CGI scripts in response to any or all URLs.

3. Write a shell CGI script which prints details of the context in which it is run. Here is an example implementation:

show_execution_context.sh.cgi (code/cgi/show_execution_context.sh.cgi)

Execution Environment
 	
pwd: /tmp_amd/kamen/export/kamen/3/cs2041cgi/public_html
id: uid=14380(cs2041cgi) gid=14380(cs2041cgi) groups=143
hostname: dvorak
uname -a: Linux dvorak 3.2.0-4-686-pae #1 SMP Debian 3.2
	<html>
	<head></head>
	<body>
	 Execution Environment
	<pre>pwd: /tmp_amd/kamen/export/kamen/3/cs2041cgi/pub
	id: uid=14380(cs2041cgi) gid=14380(cs2041cgi) groups=143
	hostname: dvorak
	uname -a: Linux dvorak 3.2.0-4-686-pae #1 SMP Debian 3.2.8
	</pre>
	</body>
	</html>

Discuss how security concerns might have affect CSE's choices for configuring how student CGI scripts are run on its web server.

CSE web server includes a mechanism to runs student CGI scripts as the student's uid.
Otherwise students could gain privileges by constructing appropriate CGI scripts.

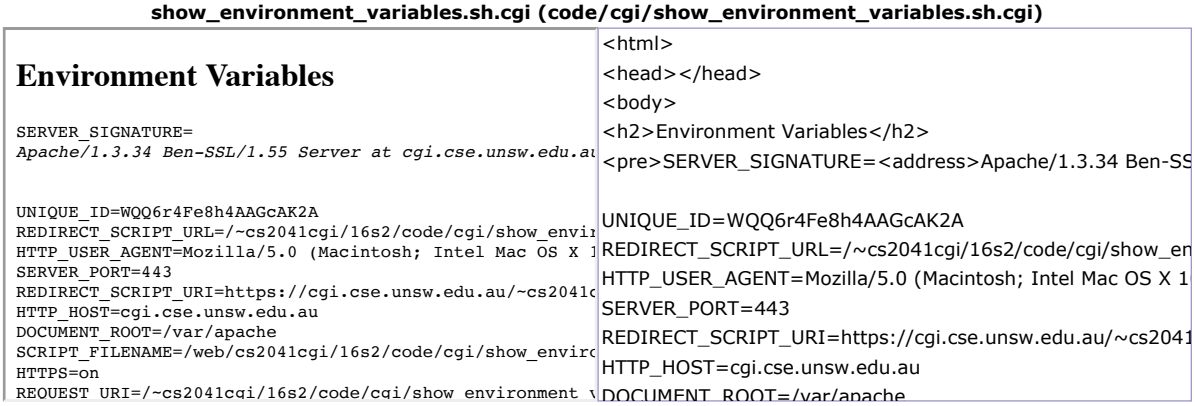
As CGI scripts are prone to externally accessible security holes. CGI script are run with only some files accessible - only as user's `public_html` is visible - mounted as `/web/user`

CGI scripts are also run on separate servers.

```
#!/bin/sh
# Simple CGI script written by andrewt@cse.unsw.edu.au
# Print some HTML plus information about the environment
# in which the CGI script has been run

cat <<eof
Content-type: text/html

<!DOCTYPE html>
<html lang="en">
<head></head>
<body>
<h2>Execution Environment</h2>
<pre>
eof

for command in pwd id hostname 'uname -a'
do
    echo "$command: `$command`"
done

cat <<eof
</pre>
</body>
</html>
eof
```

4. Web servers pass some useful information to CGI scripts as environment variables.
   Write a shell CGI script which prints details of the environment variables it has been passed. Here is an example implementation:

show_environment_variables.sh.cgi (code/cgi/show_environment_variables.sh.cgi)



Discuss what some of these environment variables might indicate.

Sample solution

```
#!/bin/sh
# Simple CGI script written by andrewt@cse.unsw.edu.au
# Print some HTML plus the environment
# passed to CGI script by the web server
# Note a < character in environment variable values will be interpreted as a HTML tag

cat <<eof
Content-type: text/html

<!DOCTYPE html>
<html lang="en">
<head></head>
<body>
<h2>Environment Variables</h2>
<pre>
`env`
</pre>
</body>
</html>
eof
```

5. A HTTP request can pass parameters to a web server. The web server passes these on to the CGI script typically in environment variable(s) for GET request and on STDIN for POST requests.
   Write a shell CGI script which prints whether it was invoked with a GET request or a POST request and prints details of the parameters it has been passed. Here is an example implementation:

show_input_parameters.sh.cgi?comp2041=best-course-ever (code/cgi/show_input_parameters.sh.cgi?comp2041=best-course-ever)

GET Request - Input Parameters

comp2041=best-course-ever

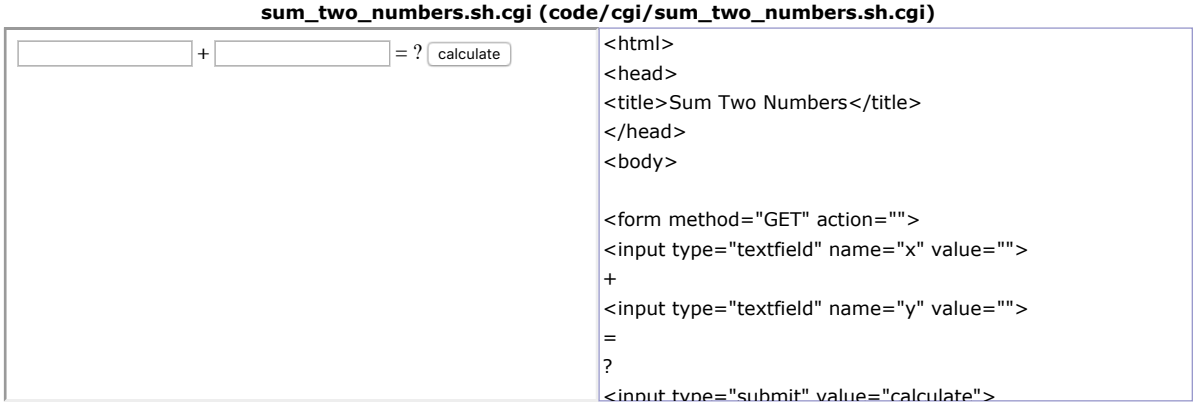
```
<html>
<head></head>
<body>

<h2>GET Request - Input Parameters</h2>
<pre>comp2041=best-course-ever
</pre>
</body>
</html>
```

Sample solution

```
#!/bin/sh
# Simple CGI script written by andrewt@cse.unsw.edu.au
# Output some simple HTML and the input parameters
# the web server has passed on to the CGI script.

if test "$REQUEST_METHOD" = POST
then
    parameters="`cat`"
else
    parameters="$QUERY_STRING"
fi

cat <<eof
Content-type: text/html

<!DOCTYPE html>
<html lang="en">
<head></head>
<body>

<h2>$REQUEST_METHOD Request - Input Parameters</h2>
<pre>
$parameters
</pre>
</body>
</html>
eof
```

6. A HTTP request can pass pararmeters to a web server. The web server passes these on to the CGI script typically in environment variable(s) for GET request and on STDIN for POST requests
   Write a shell CGI script which (given no parameters) prints a form allowing 2 numbers to be entered. The form should run the same CGI script, and the CGI script should print the sum of the numbers and a form allowing two more numbers to be entered and the script to be run again. Here is an example implementation:

sum_two_numbers.sh.cgi (code/cgi/sum_two_numbers.sh.cgi)



+ = ? calculate

```
<html>
<head>
<title>Sum Two Numbers</title>
</head>
<body>

<form method="GET" action="">
<input type="textfield" name="x" value="">
+
<input type="textfield" name="y" value="">
=
?
<input type="submit" value="calculate">
```

Sample solution

```
#!/bin/sh
# Simple CGI script written by andrewt@cse.unsw.edu.au
# Sum two numbers and outputs a form which will rerun the script


# Note removal of characters other than 0-9 . - + to avoid potential security problems

if test $REQUEST_METHOD = "GET"
then
    parameters="$QUERY_STRING"
else
    read parameters
fi

x=`echo $parameters|sed '
    s/.*x=//
    s/&.*//
    s/[^0-9\-\.\+]//g
    '`
y=`echo $parameters|sed '
    s/.*y=//
    s/&.*//
    s/[^0-9\-\.\+]//g
    '`

cat <<eof
Content-type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
<title>Sum Two Numbers</title>
</head>
<body>
eof

sum="?"
test "$x" -a "$y" && sum=`expr "$x" '+' "$y"`

cat <<eof

<form method="GET" action="">
<input type=textfield name=x value=$x>
+
<input type=textfield name=y value=$y>
=
$sum
<input type="submit" value="calculate">
</form>
</body>
</html>
eof
```