cis20.1-fall2007-sklar, EXTRA CREDIT assignment

instructions

- This is an extra credit assignment that goes along with unit III. It is worth 10 points.
- **It is due by Monday December 10**.
- Email me to let me know if you have opted to complete this assignment. I will assume that you have followed the naming instructions below, so I will look on your class server account for **bday.html** and **bday2.html**, as described below.

1 trying out perl

1. In your `public_html` directory on the class web server, create a subdirectory called `cgi-bin`.
2. Set the file permissions on the subdirectory to be: "owner everything, group and world readable and executable":
   `unix-promt$ chmod 755 cgi-bin`
3. The files you create below will go in your `public_html/cgi-bin` directory on the class web server.
4. For all the files you create, set the file permissions to be: "owner everything, group and world readable and executable":
   `unix-promt$ chmod 755 <your-file>`
5. Note that even though the files you create will be written in **perl**, they need to have the file extension `.cgi` in order to be executed as CGI programs invoked by the web browser.
6. Type in the following perl program and save it in a file called `my.cgi` in the `public_html/cgi-bin` directory on your account on the class web server.

```
#!/usr/bin/perl

print "Content-type: text/html\n\n";

print "<html><body><h1>about this server</h1><p>";
print "<p>server name: $ENV{'SERVER_NAME'}";
print "<p>port: $ENV{'SERVER_PORT'}";
print "<p>software: $ENV{'SERVER_SOFTWARE'}";
print "<p>protocol: $ENV{'SERVER_PROTOCOL'}";
print "<p>CGI: $ENV{'GATEWAY_INTERFACE'}";
print "</body></html>";
```

7. Test this program in your browser by going to: `http://146.245.250.181/~your-username/cgi-bin/my.cgi`

   If this part doesn't work, ask me for help! You CAN'T get any further with the lab if this part does not succeed!

2 learning perl

1. `hello1.cgi` *(1 point)*

   (a) Create a new file called `hello1.cgi`.

(b) Create two scalar variables, one that is initialized to your first name and one that is initialized to your last name.

(c) Add a print statement so that the script says hello to you (both first and last names, using the variables).

See the **Scalar variables** section of the perl tutorial for assistance. Note that here is where the use of double versus single quotes is important. When you use double quotes around a variable (e.g., `"$a"`), then the print statement will *interpret* or *evaluate* the variable and print out its value. If you use single quotes around a variable (e.g., `'$a'`), then the print statement will NOT interpret or evaluate the variable and will simply print $a. Try it both ways and see what happens, but make sure that you submit it the right way (printing the value)!

2. `hello2.cgi` *(1 point)*

(a) Copy your `hello1.cgi` script to a file called `hello2.cgi`

(b) Edit `hello2.cgi` as follows.

(c) Initialize two scalar variables to the NUMERIC day and month of your birthday (I won't ask for the year—you don't have to tell us how old you are...). The day should be a value between 1–31, and the month should be a value between 1–12.

(d) Add another print statement to the script so that it displays your birthday (using the variables) after it says hello to you.

3. `hello3.cgi` *(1 point)*

(a) Copy `hello2.cgi` to `hello3.cgi` and modify it as follows.

(b) Initialize an `array` variable that contains the names of the months.

(c) Change the print statement so that it prints your birthday using the month name instead of the number (again, using the variables).

(d) For example, my output would be something like this:

```
hello prof sklar!
your birthday is january 1
```

4. `hello4.cgi` *(1 point)*

(a) Copy `hello3.cgi` to `hello4.cgi` and modify it so that it calculates how many days it is from the current day until your birthday and then prints out that number.

(b) Use the built-in function `localtime()` to get the current time.

You can read about this function if you follow one of the links on the references page to **perl built-in functions**. Basically, the syntax for the function is this:
`($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)=localtime();`
Note that you call the function without an argument to get the current time. The function returns an array, so you can call it as above, or like this:
`@current_time = localtime()`
and then look up the individual array entries (e.g., `$current_time[0]`) if that makes more sense to you (the entries are stored in the order listed, i.e., seconds is item 0).

Hint: define an array that contains the number of days in each month, e.g.:
`@daysinmonth = ( 31,28,31,30,31,30,31,31,30,31,30,31 );`
You don't have to worry about leap year (2005 is not a leap year, and you don't need to write a general purpose script for this lab...).

5. `hello5.cgi` *(3 points)*

   One of the more powerful uses of perl to run CGI scripts is when they are interfaced with HTML forms.

   (a) Create an HTML form with four text fields asking for:
   - a user's first name,
   - a user's last name,
   - a user's birth month (jan-dec) and
   - a user's birth day (1-31).

   Put this in a file called `bday.html` in your `public_html` directory.

   (b) Use `method=get` to send the form variables to a CGI script called `cgi-bin/bday.cgi`, i.e.:

   ```
   <form name="bdayform" method="get" action="cgi-bin/bday.cgi">
   ```

   (c) In your `public_html/cgi-bin` directory, create a perl CGI script called `bday.cgi`.

   (d) Use the following to get the values of the form variables:

   ```
   $input = $ENV{'QUERY_STRING'}
   ```

   (e) Display the content of `$qs` to make sure you are getting the right values from the form. It should look something like this:

   ```
   fname=eliz&lname=sklar&bmonth=jan&bday=1&send=send
   ```

   (f) Now parse the content of the query string and assign variables to the user's first name, last name, birth month and birth day.

   (g) Finally, repeat the functionality of `hello4.cgi` using the values you just got from the form, instead of values that you hard-coded into your script.

6. `hello6.cgi` *(1 point)*

   (a) Create a copy of `bday.html` and call it `bday2.html`. The only difference between this one and the one from the previous step is that you should use the **post** method to send variable values from the form to the HTML script instead of the get method.

   (b) That is, use `method=post` to send the form variables to a CGI script called `cgi-bin/bday2.cgi`, i.e.:

   ```
   <form name="bdayform" method="post" action="cgi-bin/bday2.cgi">
   ```

   (c) In your `public_html/cgi-bin` directory, create a perl CGI script called `bday2.cgi` that is just like `bday.cgi`, except instead of getting the values of the form variables from the `QUERY_STRING` environment variable, you'll get them from **stdin**:

   ```
   $input = <STDIN>;
   ```

   (d) As above, parse the content of the input string and assign variables to the user's first name, last name, birth month and birth day. Display the variable values to make sure you are getting the right values from the form.

   (e) Finally, repeat the functionality of `hello5.cgi` using the values you just got from the form via stdin (instead of the query string).

7. `hello7.cgi` *(2 points)*

   (a) Extend either `bday.cgi` or `bday2.cgi` to perform error checking on agreement between the month and day number entered by the user. In other words, if the user's input indicates that their birthday is on "February 31", then your program should issue an error message.

   (b) Use **perl pattern matching** to compare the value of the month name input string and check if the number of days selected by the user is valid for that month.
   HINT: Look at the pattern matching explanation (pages 31-32) and examples (pages 33-35) in lecture III.5 notes.