

## COMP 722 E-commerce Fall 2006 Programming Assignment 5

Due in the digital drop box Monday, Nov. 27 by 11:00 PM

1. The following HTML document (with fragments missing) is in a file `prob1.pl`. The body consists only of a form with “`prob1`” as the value of its `name` attribute. The form contains the following elements in the order listed.

- A text box with “`name`” as the value of its `name` attribute. This is for the user’s name.
- Two radio buttons, both with “`gender`” as the value of their `name` attribute. This is for the user to record his/her gender. One button has “`male`” as the value of its `value` attribute. The other button has “`female`” as the value of its `value` attribute.
- A `select` element rendered as a scrolling box with “`status`” as the value of its `name` attribute and `size 2`. This gives three options with the following values: “`Undergrad`” (with rendering `Undergrad`), “`Grad`” (with rendering `Grad`), and “`Part-time`” (with rendering `Part-time`).
- A reset button that displays **See the values**.

When the reset button is clicked, function `echo()` is invoked. This checks (using a regular expression) that the value entered for the user’s name is a string of word characters, then some white space, then another string of word characters. If so, it displays these two names in an `alert` box. If not, it displays in an `alert` box a message to the effect that the name entered is in the wrong format. It then loops over the array of objects corresponding to the radio buttons, finds that one that is checked, and raises an `alert` box that identifies the user’s gender as the value of the radio button that is checked. Finally, it then displays the value selected as the student’s status in an `alert` box.

Supply the missing code. You can download this file (with the fragment missing) from the location where you got this assignment.

```

<html>
<head>
<title>Programming Assignment 5, Problem 1</title>
<script type = "text/javascript">
function echo()
{
  var theName = α_____ ;
  if ( theName.match( /(\\w+)\\s+(\\w+)/ ) )
    window.alert( RegExp.$1 + "\\n" + RegExp.$2 );
  else
    window.alert( "You entered your name in the wrong format" );

  β
  _____
  _____ ;

  window.alert( γ_____ );
}
</script>
</head>

<body>
<form name = "probl" onreset = "echo()">
<p>Your name:
  < δ _____ >
</p>

<p>Your gender:<br/>
  < ε _____ >
  Male <br/>

  < ζ _____ >
  Female <br/>
</p>

<p>Your student status:<br/>
  < η _____ >
  < θ _____ >Undergrad</option>
  < ι _____ >Grad</option>
  < κ _____ >Part-time</option>
  </select>
</p>

< λ _____ >
</form>
</body>
</html>

```

The following is how this page is rendered when it first comes up.



The screenshot shows a web form with the following elements:

- A text input field labeled "Your name:".
- Two radio buttons labeled "Male" and "Female" under the label "Your gender:". The "Male" radio button is selected.
- A dropdown menu labeled "Your student status:" with two visible options: "Undergrad" and "Grad".
- A button labeled "See the values".

- $\alpha$ : The value entered in the text box
- $\beta$ : Loop over the array of objects corresponding to the radio buttons, find that one that is checked, and raise an `alert` box that identifies the user's gender as the value of the radio button that is checked.
- $\gamma$ : The value of the option selected for the `select` element
- $\delta$ : A text box with "name" as the value of its `name` attribute
- $\epsilon$ : A radio button with name "gender" and value "male"
- $\zeta$ : A radio button with name "gender" and value "female"
- $\eta$ : A `select` element with name "status" that allows two of its options to be displayed at a time
- $\theta$ : The option with value "Undergrad"
- $\iota$ : The option with value "Grad"
- $\kappa$ : The option with value "Part-time"
- $\lambda$ : A reset button that displays **See the values**

**Two more problems to come!**

2. The first file listed below (with gaps) is an HTML document with a form. The form (see the rendering at right) has a text box with name "name" for the user's name and another, with name "age", for his age. It also has three radio buttons, all with name "payment", for the user's method of payment. The values for the radio buttons are "discover", "master", and "check". The form also has a submit button. When the form is submitted, the GET method is used to send the request to `cgi-bin/prob2.cgi`. In this listing, gaps are identified with Greek letters, which are repeated below with descriptions of what to put in the gaps. The listing of `prob2.cgi` (also with gaps) is next.

```

<html>
<head>
  <title>Assignment 5, Problem 2</title>
</head>
<body>
<form α β >
  <p>
    Your name:
    γ
  </p>

  <p>
    Your age:
    δ
  </p>

  <p>
    Method of payment:<br>
    ε
    Discover Card <br/>
    η
    Master Card <br/>
    ζ
    Check <br/>
  </p>

  <input type="submit" value="Submit">
</form>
</body>
</html>

```

$\alpha$ : Specify the GET method

$\beta$ : Specify that the request be sent to `cgi-bin/prob2.cgi`

$\gamma$ : A text box with name “name”

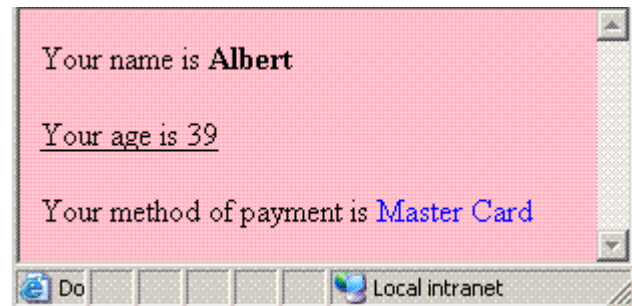
$\delta$ : A text box with name “age”

$\epsilon$ : A radio button with name “payment” and value “discover”

$\eta$ : A radio button with name “payment” and value “master”

$\zeta$ : A radio button with name “payment” and value “check”

The CGI script `prob2.cgi` (written in Perl) basically echoes back the values it receives in the request. See the rendering at right. It first assigns the value of the `payment` parameter to `$pay` then uses this in a multi-branch conditional to assign to `$pay_text` a string with the full name of the payment method (e.g., “Master Card”). It then outputs the HTML for the response, using CGI.pm methods. Arguments to `start_html` specify “Welcome” as the title and that the background color of the body be pink. The name, age, and payment values are output in `p` elements. The name is inside a `strong` element (use the `param` method inside a call to the `strong` method). The entire paragraph giving the age is underlined (use a `-style` argument). And the color of the payment method only (not the entire paragraph in which it occurs) is blue. For this, have `$pay_text` be an argument of a call to the `span` method with a `-style` argument.



Have your HTML document in the normal folder, viz., the server root `htdocs`, and have your CGI script in the normal folder, viz., the `cgi-bin` folder that is a sibling of `htdocs` (but is aliased so as to be addressed as a child of the server root). The listing below includes the pragma

```
use CGI::Carp qw( fatalToBrowser );
```

so that compilation and runtime errors are reported in the browser window (see sec. 21.3 of the Course Notes). The listing also includes the pragma

```
use CGI::Pretty qw( -debug );
```

Exploit this as described in sec. 21.5.1: execute the script in the command window, redirecting the output to a file and supplying `name=value` pairs for the parameters after issuing the `perl` command.

Submit your solution in the usual way. I will assume that your files reside in folders as described above.

```
#!/C:/Perl/bin/perl

use strict;
use CGI::Pretty qw( -debug );
use CGI::Carp qw( fatalsToBrowser );

my $q = new CGI;

my $pay = α;
my $pay_text;

if ( $pay eq "discover" ) {
    $pay_text = "Discover Card";
}
elsif ( $pay eq "discover" ) {
    $pay_text = "Master Card";
}
elsif ( $pay eq "check" ) {
    $pay_text = "by check";
}
else {
    $pay_text = "unknown method";
}

print $q->header,

    $q->start_html( β,
                  γ ),

    $q->p( "Your name is ",
         δ ),

    $q->p( { ε },
         "Your age is ", η ),

    $q->p( "Your method of payment is ",
         ζ,

    $q->end_html;
```

α: The value of parameter “payment”

β: Specify the title “Welcome.”

γ: Specify a pink background color for the body.

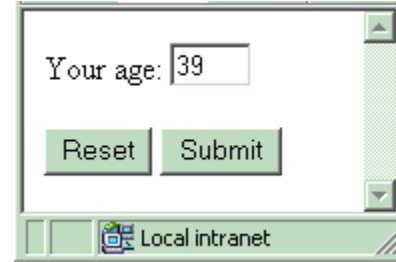
δ: Put the value of parameter “name” in a strong element.

ε: Specify that the paragraph is underlined (using `-style`).

η: The value of the “age” parameter

ζ: Call the `span` method with `$pay_text` as an argument and specifying blue text.

3. This problem is similar to the first example in section 21.5.7 in the Course Notes but is considerably simpler. An HTML document (listed below with gaps and whose rendering is shown at right) contains a form (named “prob3”) whose sole element, besides a reset and a submit button, is a textbox named “age” for the user’s

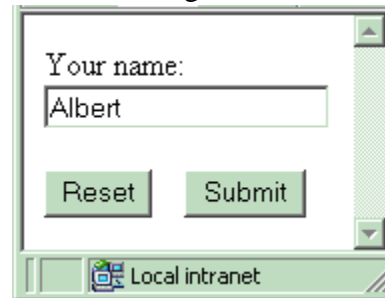
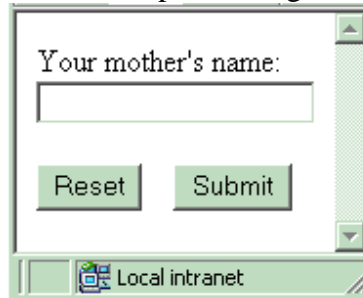


age. The function `validate()` is a listener for the `submit` event and checks that the text box contains one to three digits. If not, submission is aborted and a wrong-format message is displayed in an alert box, as shown at right. Also, the content of the text box is selected (causing it to appear in reverse video, which does not show up until the alert box is dismissed). (Note that this function gives the user another chance.)

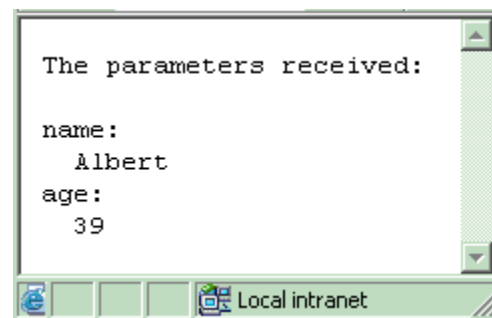


The CGI script `prob3.cgi` (also listed below with gaps) returns as its response an HTML document that also contains a form. One element in this form is a hidden field named “age” whose value is the age value sent when the original form was submitted. What other element is included in the form depends on the value of the age parameter. If this is less than eighteen, then a textbox named “motherName” is included, and the rendering is as shown in the first screen capture at right. If, however, the age is at least

eighteen, then a textbox named “name” is included, and the rendering is as in the next screen capture at right. When this form is submitted, a request is sent to the script `param_list.cgi`, which is one of the example files for Part 21; its plain-text response simply lists the names and values of the parameters it received. For example, when values are entered as shown in the first and fourth screen captures above, the rendering of the content in the response produced by `param-list.cgi` is as shown at right.



The HTML document (`prob3.html`) and the script `prob3.cgi` can be downloaded from where you got this assignment. The .zip file also includes `param_list.cgi`. Submit your solutions in the usual way. I assume that the HTML document resides in the server root, `htdocs`, and that the CGI scripts reside in the `cgi-bin` folder aliased to be a child of `htdocs` (the default set-up).



The following is the listing of `prob3.html`. Gaps are labeled with Greek letters, which are repeated after the listing with descriptions of the code that should go in the corresponding gaps.

```

<html>
<head>
  <title>Programming Assignment 5, Problem 3</title>
  <script language="javascript">
    function start()
    {
      α_____i;
    }

    function validate( event )
    {
      if ( β_____ {
        γ_____i;
        δ_____i;
        alert( "Wrong format for age!" );
      }
    }

  </script>
</head>
<body onload="start()">
<form ε_____
  _____>
  <p>
    Your age:
    ζ_____
  <p>

  <input type="reset">
  <input type="submit" value="Submit">
</form>
</body>
</html>

```

$\alpha$ : Make the function `validate` a listener for the submit event (for the form element).

$\beta$ : An expression that is true if the value of the textbox does not contain one to three digits. (**Hint**: Invoke the `match()` method of the value of this textbox, passing it the appropriate regular expression.)

$\gamma$ : Suppress the form submission.

$\delta$ : Select the text box.

$\epsilon$  (Occupies parts of two lines in the listing): Name the form “`prob3`” and specify that it use the POST method to send the request to `cgi-bin/prob3.cgi`.

$\zeta$ : A textbox named “age” that is three characters wide.

