Program 3: PHP Error Checking

Student Learning Objectives

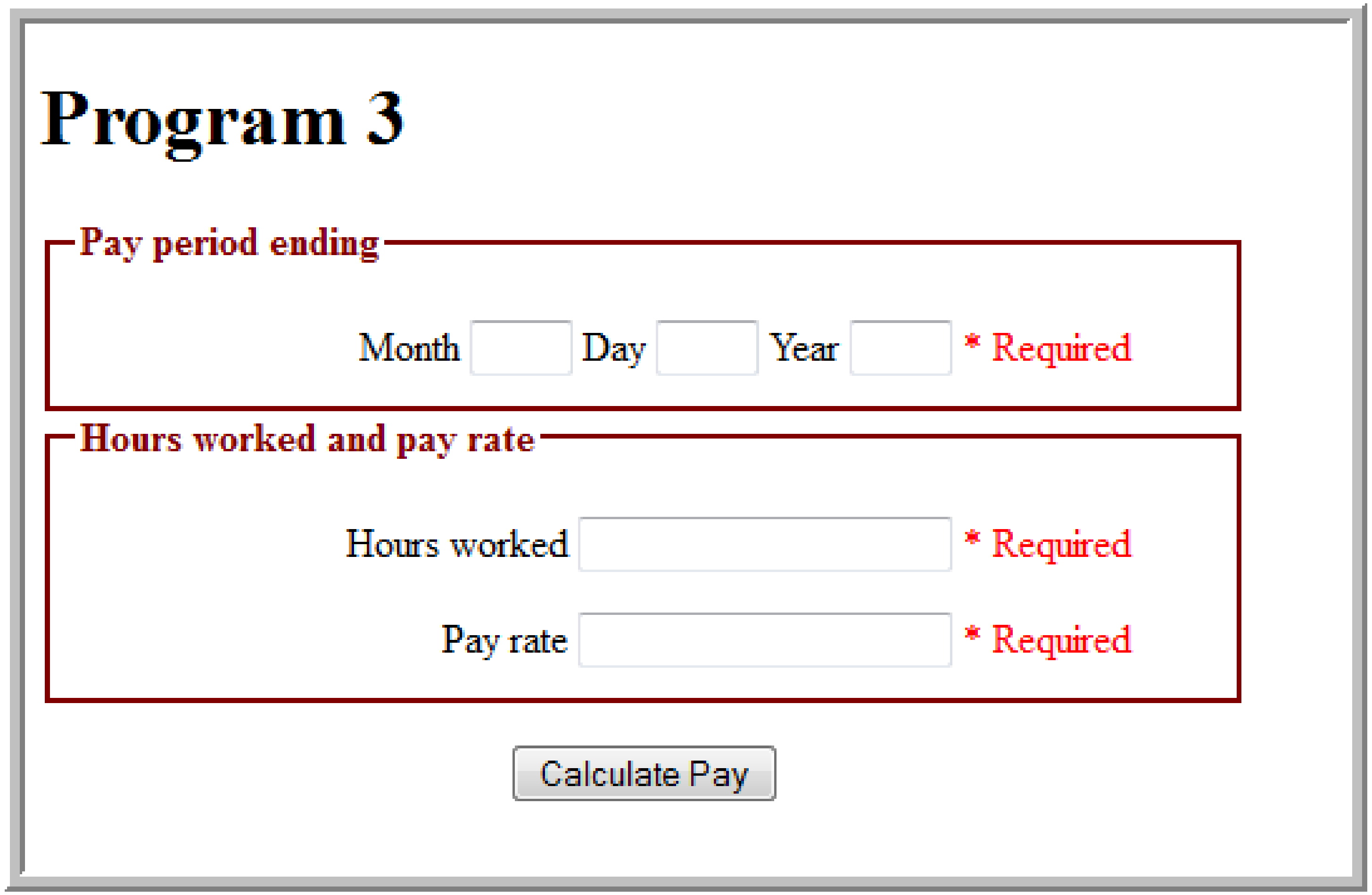
- Demonstrate the ability to author XHTML forms and tables
- Demonstrate the ability to author valid externally linked cascading style sheets (CSS)
    - **All** style rules should be placed in one linked CSS file, not in XHTML
- Demonstrate the ability to use PHP error checking for dates and numbers
- Demonstrate the ability to create valid,accessible, and semantically correct XHTML code

`program3.php`

This assignment tests your understanding of XHTML, CSS, and PHP. You will create a web page `program3.php` containing an XHTML form in folder `U:\htdocs\programs\program3\`. program3.php will be an ***all-in-one*** web page that

- presents the payroll form shown below with **no** default values filled in on first page load
- performs input validation for all user input according to the rules below
  - if any of the user input data is incorrect, display appropriate error messages and allow the user to fix their errors
    - re-populate the form with any previous values the user entered so they do not need to retype everything
- if the data validates correctly, `program3.php` saves the user input in session variables, redirects the server to `getPay.php`. getPay.php will use the session variables to calculate and display the user's pay

`program3.php`: **no default values**


Program 3

Pay period ending

Month [ ] Day [ ] Year [ ] * Required

Hours worked and pay rate

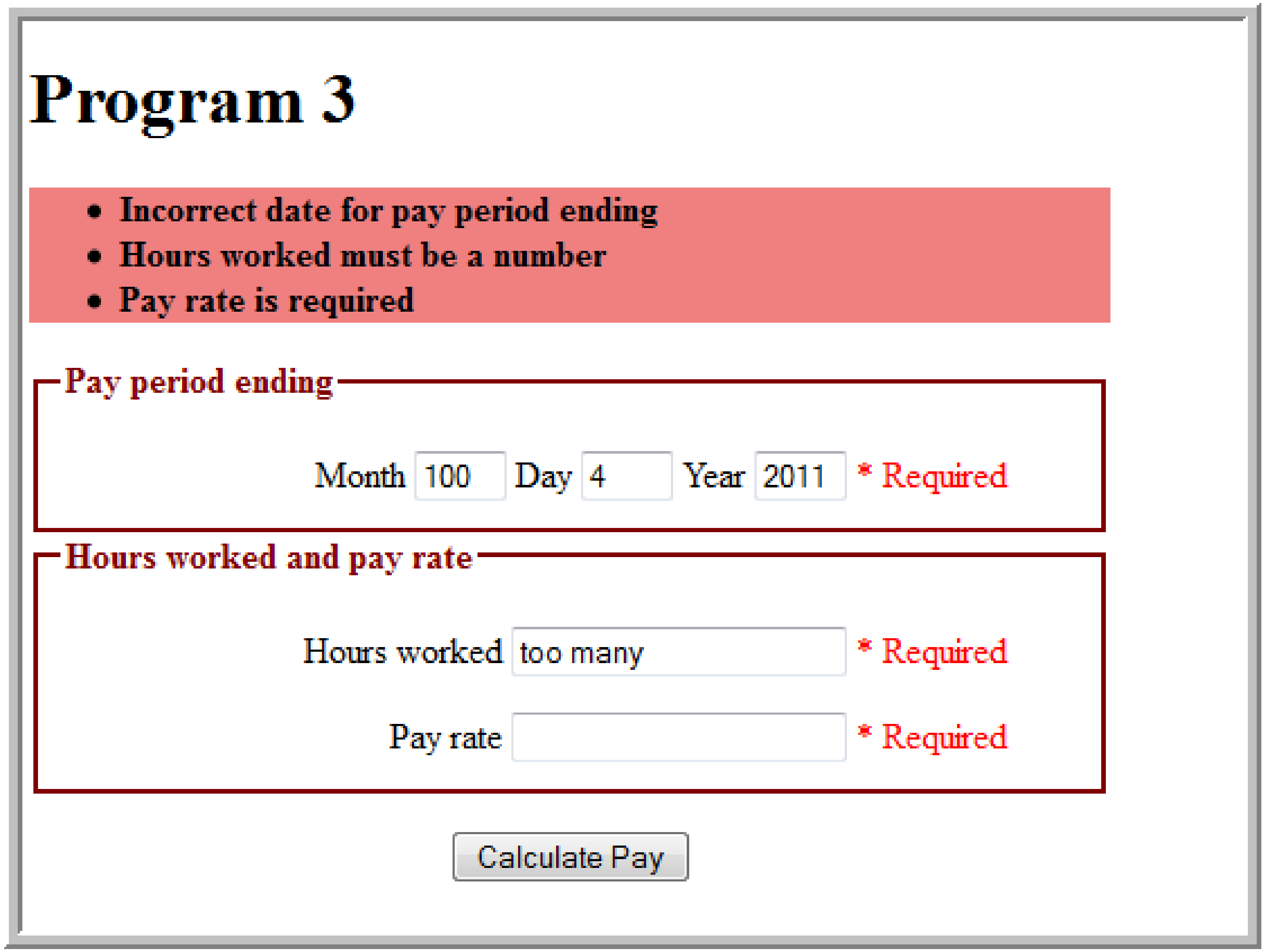
Hours worked [ ] * Required

Pay rate [ ] * Required

`program3.php` **style rules**

- Upon first load, form elements do **not** have default values
- Use **CSS** for styling and positioning (no tables in program3.php)
  - Make the form and error messages box `30em` wide
  - Display the fieldset border and legend in Maroon (#800000) with a bold font
  - Display the "* Required" text in Red
  - Study the snapshot to duplicate the other style rules so your output *approximately* matches the screen snapshot

program3.php: sample error messages and form elements retain the user input data



Program 3

- Incorrect date for pay period ending
- Hours worked must be a number
- Pay rate is required

Pay period ending

Month	100	Day	4	Year	2011	* Required

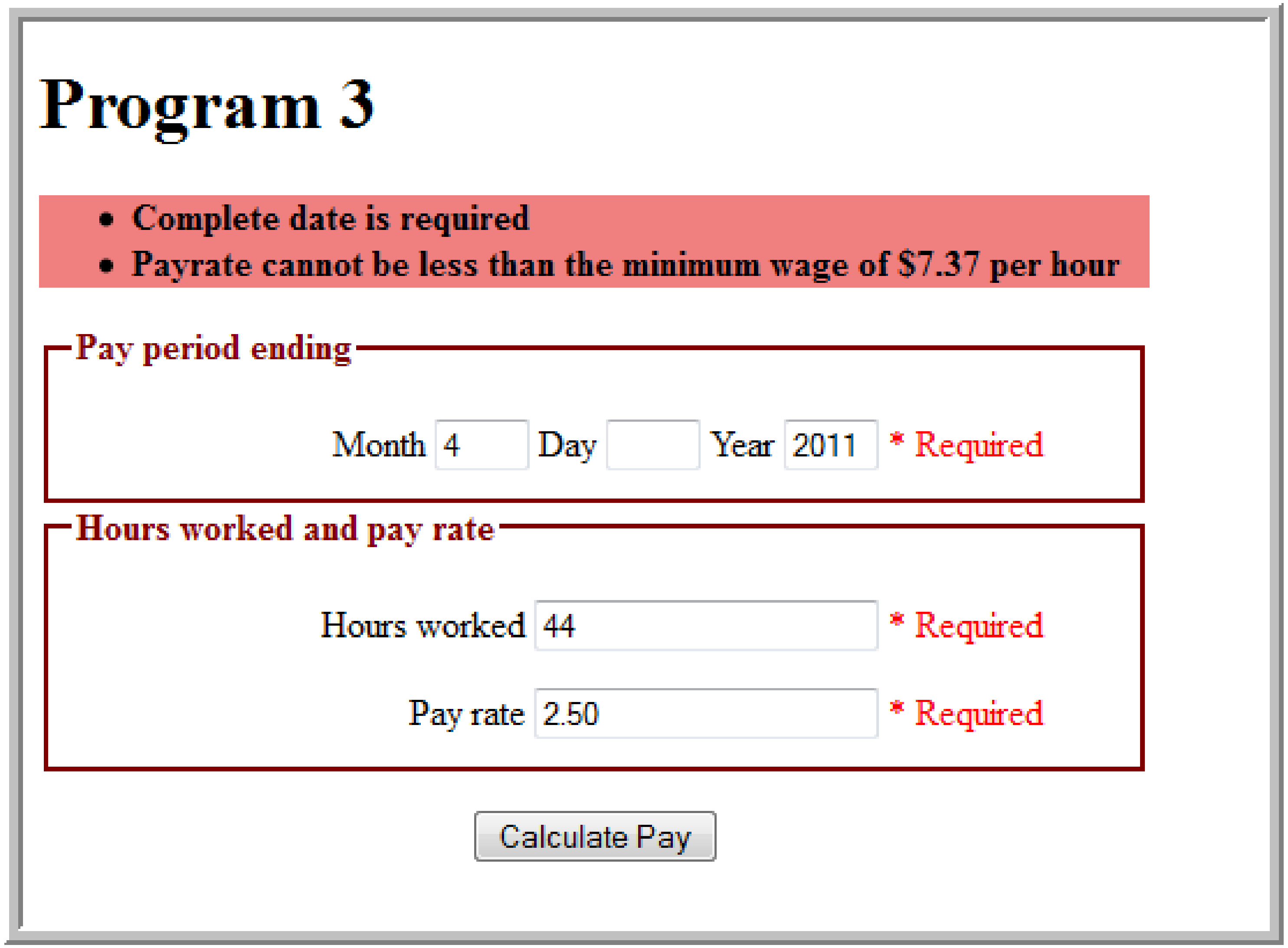
Hours worked and pay rate

Hours worked	too many	* Required
Pay rate		* Required

program3.php: Possible Error Messages

1. Complete date is required
2. Hours worked is required
3. Pay rate is required
4. Incorrect date for pay period ending
5. Hours worked must be a number
6. Pay rate must be a number
7. Hours worked must be less than the number of hours in a week
8. Hours worked must be positive
9. Payrate cannot be less than the minimum wage of $7.37 per hour

`program3.php`: more sample error messages and form elements retain the user input data



Program 3

- Complete date is required
- Payrate cannot be less than the minimum wage of $7.37 per hour

Pay period ending

Month	4	Day		Year	2011	* Required

Hours worked and pay rate

Hours worked	44	* Required
Pay rate	2.50	* Required

`program3.php`: validation tips

1. Research and use PHP's date/time functions to validate that the date entered is correct
2. Research and use PHP's variable handling functions to validate that hours and payrate are numbers

`program3.php` style rules

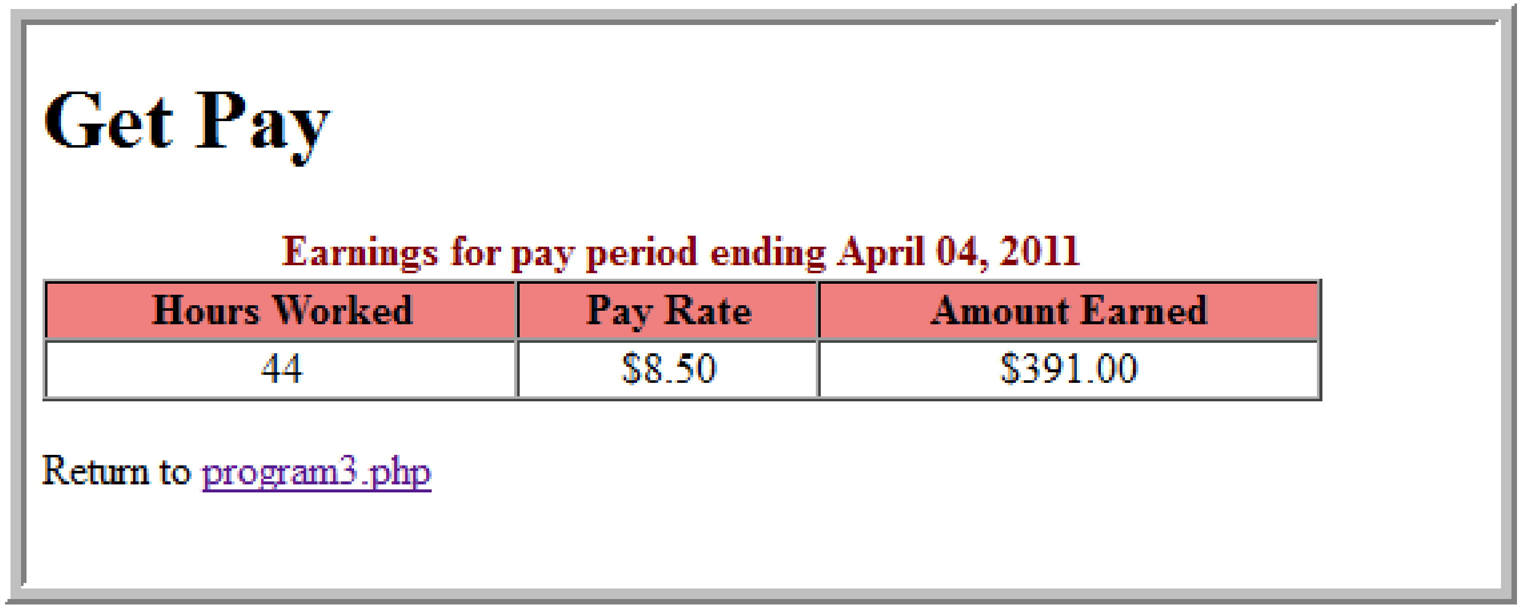
1. Re-display the user input (possibly incorrect) in the input elements so the user does not need to re-enter all values each time
2. Display error messages as an unordered list
3. Make the error messages box `30em` wide with a light coral (**#F08080**) background
4. Study the snapshot to duplicate the other style rules so your output *approximately* matches the screen snapshot

---

`getPay.php`

When the user input data validates on program3.php, use the PHP `header()` function to redirect the server to `getPay.php` to calculate and display the user's pay for one week.

- Use **session variables** to pass user data from `program3.php` **to** `getPay.php`
- Display the session variables and calculated amount earned is a neatly formatted table
  - Assume data in session variables has been validated in `program3.php`
  - Remember that any hours worked over 40 for the week are paid overtime



Get Pay

Earnings for pay period ending April 04, 2011

Hours Worked	Pay Rate	Amount Earned
44	$8.50	$391.00

Return to program3.php

`getPay.php` style rules

- Use the same **CSS** file for styling and positioning
- Make the table `30em` wide
- Display the table caption in Maroon (#800000) with a bold font
- Make the table header background color light coral (**#F08080**)
- Study the snapshot to duplicate the other style rules so your output *approximately* matches the screen snapshot

Tips

- Research and use PHP's date/time functions to display the date using the month name, 2-digit day, comma, and four digit year
- Display Pay Rate and Amount Earned as currency

`program3.css`

Use one `program3.css` linked style sheet to apply formatting to both web pages as shown in the screen snapshots

---

Grading Criteria:

Note: Program assignments are **individual assignments** and should represent your own work. You should **not** use pair programming on the program assignments. If you received help with this assignment, acknowledge this help (who helped and the nature of the help) in your file's header comment.

Validate your generated HTML code (program3.php and getPay.php) as XHTML 1.0 Strict with no errors or warnings. They should also be checked for accessibility using the FAE Rule Set with no issues identified. Do **not** express stylistic information in the HTML page itself, such as inline styles or presentational HTML tags such as `b` or `font`.

Express all stylistic information on the page using **CSS** defined in `program3.css`. For full credit, your style sheet must successfully pass the W3C CSS validator with no errors or warnings. You only need to worry about the appearance of your page in Firefox. Your pages will not be tested using Microsoft Internet Explorer or other browsers.

Format your HTML, CSS, and PHP nicely so that it is readable following the **CS 250 coding standards**. Place a comment header in each file containing your account, file, and honor code. Include metadata on your HTML pages.

To Receive Credit

- A **scoring rubric** for program 3 will be used for grading purposes.
    - Turn in the **scoring rubric** [PDF] for grading purposes
        - Keep track of and include the **total completion time** (rounded to closest half-hour) it took to complete the program assignment and include the time on the rubric
        - Write a score `0..4` in the rubric's **self assessment column** representing your completion status
    - Make sure your work is stored in your CS 250 account in folder `U:\htdocs\programs\program3\`

Program 3 is due at the start of lab on Monday. **No** late programming assignments will be accepted without prior approval.