

CS 142 Final Examination

Fall Quarter, 2010

You have 3 hours (180 minutes) for this examination; the number of points for each question indicates roughly how many minutes you should spend on that question. Make sure you print your name and sign the Honor Code below. During the examination you may consult two double-sided pages of notes as well as a solution for Project #5; all other sources of information, including laptops, cell phones, etc. are prohibited. If there is a trivial detail that you need for one of your answers but cannot recall, such as the name of a particular CSS attribute or Ruby library method, you may ask the course staff for help.

I acknowledge and accept the Stanford University Honor Code. I have neither given nor received aid in answering the questions on this examination.

(Signature)

(Print your name, legibly!)

(email id, for sending score)

Problem	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	Total
Score												
Max	12	15	4	4	8	20	14	8	15	30	35	165

Problem 1 (12 points)

Indicate whether each of the following statements is True or False, and explain your answer briefly.

- (a) In HTML *entities* are used to incorporate external content into a Web page, such as images.

- (b) A single partial-page template in Rails can be used in more than one Web page.

- (c) When a user places an order via the Web and posts the final “Confirm” form for the purchase, the Web server should normally return HTML for the order confirmation page in the response for the HTTP request.

- (d) Suppose that an online retailer uses a relational database to keep track of orders, and one table contains a row for each order. If an order contains several items, the foreign keys for each of those items would typically be stored in the row for that order.

- (e) In a scripting language types are typically associated with values, not variables.

- (f) Web browsers are stateless: if a browser isn’t displaying a page for a particular server it need not retain any state related to that server.

Problem 2 (15 points)

(a) List 3 ways in which Web applications are revolutionary compared to traditional applications.

(b) Explain 2 different ways that CSS supports the DRY (Don't Repeat Yourself) principle.

(c) Give 2 examples of how Rails takes advantage of meta-programming facilities in the Ruby language.

(d) In your opinion, which programming language is better, Ruby or Javascript? List 2 specific reasons to justify your choice.

(e) Describe a simple attack that could be executed if browsers did not implement the Same-Origin Policy.

Problem 3 (4 points)

Consider the following security attack where an evil server attempts to impersonate a good server:

- The evil server obtains the certificate for the good server (available publically from certificate authorities).
- The evil server uses an active network attack to arrange for traffic from a particular browser intended for the good server to be sent to the evil server instead.
- When a client attempts to open an HTTPS connection to the good server, the evil server receives the connection open request and behaves just like the good server, returning the good server's certificate at the appropriate point in the protocol.

Assuming that the evil server can obtain a valid certificate and mount an active network attack, can the client's security be compromised? Explain your answer.

Problem 4 (4 points)

How many alert dialogs will the following Javascript generate, and what will be displayed in each of them?

```
var x = "10";
function f(){
    var x = "4";
    alert(this.x);
    function g(){alert(x);}
    g();
}
f();
```

Problem 5 (8 points)

Below you will see some snippets of HTML from a Web page. Fill in the body of the Javascript function `changeColor` so that the color of the text changes in response to the selection made in the menu.

```
<style type="text/css">
  .a {color:red;}
  .b {color:green;}
  .c {color:blue;}
</style>
...
<div id="colorText">Select below to change the color of this text</div>
  <select onchange="changeColor(this.value)">
    <option value="a">Red</option>
    <option value="b">Green</option>
    <option value="c">Blue</option>
  </select>
  <script type="text/javascript">
    //
      function changeColor(value) {

}
//]]&gt;
&lt;/script&gt;</pre></div><div data-bbox="483 936 512 953" data-label="Page-Footer"><p>-5-</p></div>
```

Problem 6 (20 points)

Write Ruby code for a method `link` that behaves somewhat like the Rails `link_to` method. The function takes 2 arguments; the first argument is the text that should be displayed for a link (which could have come from an unknown source such as a user) and the second argument is a hash describing the URL for the link. The function returns valid XHTML for an `<a>` element. For example, consider the following invocation:

```
link_to("Click on me", :controller => "student", :action => "show",
       :name => "Lee");
```

This will return the following XHTML:

```
<a href="/student/show?name=Lee">Click on me</a>
```

Here are some additional details and requirements:

- You can assume that the second argument always contains `:controller` and `:action` entries, which describe the hierarchical portion of the URL.
- The second argument may contain any number of additional entries, such as `:name` in the example above. All of these entries should be included in the URL as query values; the names and values of these entries are of unknown origin (i.e., they could have been typed by a user).
- You may not use the Rails methods `link_to`, `url_for`, or anything similar in your solution.
- You may use the Rails method `h` in your solution, as well as the Ruby method `URI.escape`, which takes a string argument and returns a URL-encoded result (it will escape any characters other than `A-Z`, `a-z`, `0-9`, or any of `-_.` using `%`-notation).

Problem 7 (14 points)

Given the following pairs of potential attacks and countermeasures, state whether the countermeasure is ineffective, somewhat effective, or very effective against the attack, and justify your answer:

a) Spoofed sites for phishing : extended validation certificates

b) Session hijacking via stolen cookies : HTTPS

c) Reflected XSS : same-origin policy

d) Mixed content attack : relative resource links

e) Cross-site request forgery (CSRF) : HTTPS

f) SQL injection : escaping user-supplied data when generating HTML

g) Stored XSS : escaping user-supplied data when generating HTML

Problem 8 (8 points)

Label each of the tasks below with “Model”, “View”, or “Controller” to indicate where that task would typically be implemented in a Web application using an MVC architecture.

- (a) Validate form data

- (b) Make sure a user is logged in

- (c) Invoke the `link_to` method

- (d) Return a “redirect” to the browser

- (e) Define an event handler for a custom form element

- (f) Generate a new session token

- (g) Invoke the `find_all_by_name` method

- (h) Create a “salt” for a password

Problem 9 (15 points)

Write Ruby code that will define a method `partition` in the `Enumerable` module. The `partition` method invokes a block on each element of its enumerable and returns two arrays, the first containing all of the elements for which the block evaluated to true, and the second containing all of the other elements. For example,

```
[1, 5, 96, 43, 88, 17].partition {|i| (i&1) != 0}
```

should return

```
[[1, 5, 43, 17], [96, 88]]
```

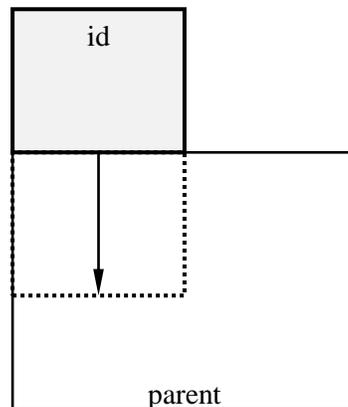
You may not use any existing methods of the `Enumerable` module in your solution.

Problem 10 (30 points)

Write a Javascript function `peekDown` that will animate the appearance of an HTML element within its parent. The function will be invoked as follows:

```
peekDown(id, duration)
```

`id` is the HTML identifier of an element, and `duration` indicates how long the animation should take, in milliseconds. `peekDown` should initially position the given element just above the upper left corner of its parent as shown in the figure below; the element will not be visible because it is entirely outside the bounds of its parent. Then `peekDown` should gradually slide the element down; as it does this, the bottom part of the element will become visible underneath the top edge of the parent. Eventually, `duration` ms later, `peekDown` should stop, leaving the entire element just visible at the top of its parent (dotted line in the figure). The result is an animation where the element “peeks” in from the top of its parent.



Additional notes and requirements:

- In order to get full credit, `peekDown` must not use any global variables (except for the `peekDown` function itself) and it must be able to support multiple simultaneous transitions.
- You can assume that the element and its parent have been created already and configured so that the child element is clipped by the boundaries of its parent as described above (CSS “`overflow: hidden;`”); all you need to do in your code is to move the child to create the animation effect.
- You can assume that neither the element nor its parent uses padding, spacing, or a border.

- You may find the following Javascript functions useful:

```
setTimeout(funcOrCode, ms);  
id = setInterval(funcOrCode, ms);  
cancelInterval(id);
```

`setTimeout` returns immediately but arranges for `funcOrCode` to be executed (exactly once) `ms` milliseconds in the future; `funcOrCode` can be either a function to invoke or a string to eval. `setInterval` is similar to `setTimeout` except that `funcOrCode` is executed repeatedly every `ms` milliseconds; `setInterval` returns a token that can be passed to `cancelInterval` to stop the repeating execution.

Working space for Problem 10:

Problem 11 (35 points)

In this problem you will extend your solution for Project #5 (Forms and Validation), the code for which you should have brought to the exam. Following the instructions below, modify the application to provide an additional URL that displays all of the comments written by a particular user (regardless of photo) sorted chronologically with the most recent comment first. For each comment the page should display the text of the comment, the date/time when the comment was created, and the name of the user who owns the photo for which the comment was made (you do not have to display the actual photo). There should also be a link that will go to the page displaying the photo (`/pics/user/id` for the photo's user).

- (a) (5 points) Describe the structure of the URL you propose to use for the new comment page.
- (b) (5 points) Describe how the new page will integrate into the application: where will there be links to the page and what form will they take? It must be easy to get to the comments for any given user.
- (c) (25 points) Modify the application to add the new functionality:
- Write all of your new code in the space below and on the next page, but indicate where in the original code each addition/modification should occur (you can mark the original code to indicate insertion points).
 - If you need new files, indicate the names for those files below.
 - Your solution must be complete and fully functional, except that you do not need to write any CSS for this problem; you can assume the CSS will be written by someone else.
 - **TURN IN THE ORIGINAL PROJECT CODE WITH YOUR EXAM.**

Working space for Problem 11: