CS 142 Midterm Examination
Spring Quarter 2012

You have 1.5 hours (90 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 4; 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)

SOLUTION SET
(Print your name, legibly!)

_____
(email id, for sending score)

Problem	#1	#2	#3	#4	#5	#6	Total
Score							
Max	12	4	4	20	10	30	80

Problem 1 (12 points)

Indicate whether each of the following statements is true or false, and explain your answer *briefly*.

(a) HTML allows constructs that are not permitted in XHTML.

Answer: TRUE. For example, you can omit close tags for many elements, or use characters such as "<" without escaping them.

(b) Once a Web server returns a cookie to a browser, the cookie will be included in all future requests from the browser to the same server.

Answer: FALSE. The cookie will normally be returned, but cookies have expiration dates, and they won't be returned once they expire.

(c) The following Ruby code will execute without errors:

```
x = 39
x = "Hello, world"
x = {:height => 42, :width => 53}
```

Answer: TRUE. Ruby is dynamically typed, which means any variable can hold any type of value, and it can hold different types at different times.

(d) In a table of a relational database, different rows can contain different numbers of fields.

Answer: FALSE. The number of fields for a table is fixed when the schema is defined and must be the same for all rows in each table.

(e) In a Web application based on MVC, if an "M" and a "C" need to communicate, it's always the "C" that calls the "M" and not vice versa.

Answer: TRUE. Models only handle the application's data, so there shouldn't be any need for them to call the controller. The controller provides the glue between models and views.

(f) A construct of the form `<% ... %>` in a Rails template can cause HTML to be generated for the resulting Web page.

Answer: FALSE. The results of the Ruby expression in the construct will not be inserted into the HTML (since it started with "<%", not "<%="), but Ruby code in the construct can invoke methods such as `concat` and `render`, which will output information to the HTML.

Problem 2 (4 points)

Describe two ways in which CSS embodies the DRY principle (Don't Repeat Yourself)

Sample answers:

1. With CSS you can define style attributes that apply to all elements of a particular class, so the details of the style need not be repeated on each element.

2. CSS stylesheets can be used in many different pages of an application.

3. CSS supports inheritance, so style attributes defined on one element will automatically apply to its descendants.

Problem 3 (4 points)

Consider the following Ruby code:

```
def method1
  x = 11
  method2 do |x|
    puts x
  end
end

def method2
  x = 22
  yield 33
end

def method3
  x = 11
  method2 do |y|
    puts x
  end
end
```

(a) What output (if any) is generated when `method1` is called?

Answer: 33.

(b) What output (if any) is generated when `method3` is called?

Answer: 11

Problem 4 (20 points)

For large Web sites CSS information is typically broken up into numerous stylesheets, where different stylesheets are used for different elements of the Web site. For example, each partial page template might have an associated stylesheet with the CSS relating to that partial. A given page will typically need to use multiple stylesheets, depending on which elements are included in the page.

In this problem you must implement a general-purpose mechanism for including stylesheets in Web pages generated using Rails. Templates and partials will invoke the method `need_stylesheet` for each stylesheet that they need:

```
<% need_stylesheet "foo" %>
```

This indicates that the stylesheet `foo.css` (located in the directory `app/assets/stylesheets`) should be included in the current Web page. In the process of rendering a view, this method may be called multiple times, and the same stylesheet may be requested in multiple places (a partial that invokes `need_stylesheet` might be used multiple times in the same page). You must write the code for `need_stylesheet`, plus make related modifications to the application layout, in order to ensure that each Web page contains exactly one `<link>` element for each requested stylesheet. There must be no duplicates, and your results must be valid XHTML.

(a) (10 points) Write Ruby code that defines the `need_stylesheet` method. You can assume that `need_stylesheet` is a global method in the file `app/helpers/application_helper.rb`, so it will be available in every view and partial page template.

Answer: keep track of all the requested stylesheets in a hash (or set, or array):

```
def need_stylesheet(name)
  if !@sheets then
    @sheets = new Hash
  end
  @sheets[name] = 1
end
```

Problem 4, cont'd

(b) (10 points) Modify the application layout below to work properly with `need_stylesheet` (you can assume that this layout will be used for all Web pages that invoke `need_stylesheet`).

```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 4  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 5    <head>
 6      <title><%= @title %></title>
 7    </head>
 8    <body>
 9      <%= yield %>
10   </body>
11 </html>
```

Answer: insert after line 6:

```
<% if @sheets then
     @sheets.each do |name|
       concat stylesheet_link_tag(name)
     end
   end
%>
```

Problem 5 (10 points)
You have been hired to work on the Web site for FootBook, a hot new Web startup that is creating social networks related to shoes. The company's database contains two tables, one that keeps track of all the users, and one that maintains friendship relationships between users:

```
users:
+----+-----------+--------------+------+-----------+
| id | name      | phone        | age  | shoe_size |
+----+-----------+--------------+------+-----------+
|  1 | Anderson  | 650-943-8027 |  19  |    8B     |
|  2 | Jones     | 408-322-4339 |  43  |    11D    |
|  3 | Hernandez | 650-715-1414 |  31  |    6A     |
|  4 | Chen      | 925-622-0368 |  28  |    3B     |
|... | ...       | ...          |  ... |           |
+----+-----------+--------------+------+-----------+

friends:
+-----------+------------+
| first_id  | second_id  |
+-----------+------------+
|         1 |          2 |
|         1 |          3 |
|         2 |          4 |
|         3 |          7 |
|       ... |        ... |
+-----------+------------+
```

For each friendship there is exactly one entry in the friends table, containing the primary keys in `users` of the two friends. Write an SQL query that will return the names of all users who have friends older than they are. Start your query with "`SELECT DISTINCT`": the `DISTINCT` keyword will ensure that duplicate names are suppressed, so each name appears at most once.

Answer:

```
SELECT DISTINCT u1.name
FROM users u1, friends, users u2
WHERE (u1.id = friends.first_id AND u2.id = friends.second_id)
OR (u2.id = friends.first_id AND u1.id = friends.second_id)
AND u1.age < u2.age
```

Problem 6 (30 points)

In this problem you will extend your solution for Project 4 (Migrations and Models), the code for which you should have brought to the exam. The overall goal is to extend the solution so that your application can keep track of the users present in each photo and display that information next to the photo.

- 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 do not need to turn in your original project code; just indicate where in the original code you would make the changes.
- If you need to create new files, indicate the names for those files below.

(a) (18 points) Modify the Project 4 solution so that it can store information about the users that appear in each photo. "Users" refers to people that are in the `users` table of your application. You must be able to handle any number of people in each photo. You do not need to worry about entering actual data into the database; you just need to set up the correct application structure to manage this information.

Answer:

Run "rails generate migration create_tags" and put the following code in the new file created in app/db/migrate:

```
class CreateTags < ActiveRecord::Migration
  def up
    create_table :tags do |t|
      t.column :photo_id,        :integer
      t.column :user_id,         :integer
    end
  end

  def down
    drop_table :tags
  end
end
```

Create app/models/tag.rb:

```
class Tag < ActiveRecord::Base
  belongs_to :user
  belongs_to :photo
end
```

Add the following line to app/models/user.rb:

```
has_many :tags
```

Add the following line to app/models/photo.rb:

```
has_many :tags
```

(b) (12 points) Modify the Project 4 solution so that the names of the users in each photo are displayed next to the photo. You do not need to write any CSS for this problem; you can assume that the CSS will be written by someone else.

Answer: add the following code to app/views/pics/user.html.erb (exact location will depend on the structure of your view, "photo" is assumed to be the current photo being rendered):

```
<div class="tagged_users">
People in photo:
<% photo.tags.each do |tag| %>
   <div class="tagged_user">
     <%= tag.user.first_name %> <%= tag.user.last_name %>
   </div>
<% end %>
</div>
```