COMP/EECE 7012

Exam 1

Spring 2015

Name: _____, _____

Last name First name

Rules:

- No potty breaks.
- Turn off cell phones/devices.
- Closed book, closed note, closed neighbor.

- WEIRD! Do not write on the backs of pages. If you need more pages, ask me for some.

Reminders:

- Verify that you have all pages.
- Don't forget to write your name.
- Read each question carefully.
- Don't forget to answer every question.

1. [6pts] Imagine a Git repository ("repo") with one commit and one branch (master). A user makes a new commit to the repo. Draw a box-and-line diagram of the repo like the diagrams shown in class. Include all commit nodes (with made-up hashes), all HEAD nodes, and all branch nodes.

2. [6pts] Continuing the previous scenario, the user creates and checks out a new branch "my-branch" and then makes a new commit. Update the previous diagram to reflect these actions.

Consider these four versions of a Rails model file, *cat.rb*:

Version A

```
class Cat < ActiveRecord::Base
end
```

Version B

```
class Cat < ActiveRecord::Base
  validates :pet_name, length: { maximum: 30 }
end
```

Version C

```
class Cat < ActiveRecord::Base
  validates :pet_name, length: { maximum: 60 }
end
```

Version D

```
class Cat < ActiveRecord::Base
<<<<<<< HEAD
  validates :pet_name, length: { maximum: 60 }
=======
  validates :pet_name, length: { maximum: 30 }
>>>>>>> alice-branch
end
```

Now, imagine the following scenario. Alice and Bob are using GitHub to collaborate on a Rails project. They both are working in parallel on Version A of *cat.rb*. Each modifies the file *cat.rb* as follows: Alice changes it to be Version B, whereas Bob changes it to be Version C (note the difference in length maximum).

Bob commits his changes first, and pushes them to the GitHub repo. Then, Alice commits her changes.

3. [6pts] Which one of the following would happen if Alice next did a Git **push**?

    a. Her commit would be added to the GitHub repo, making the current version in GitHub be Version B

    b. Her commit would be merged with Bob's in the GitHub repo, making the current version in GitHub be Version D

    c. Her push would be rejected, leaving the current version in GitHub as Version C

    d. Her push would be rejected, leaving GitHub unchanged, and Bob's code would be downloaded and merged into her working directory, making her working copy Version D

    e. None of the above

4. [6pts] Which one of the following would happen if Alice instead did a Git **pull**?

    a. Her pull would be rejected, leaving her working directory with Version B of *cat.rb*

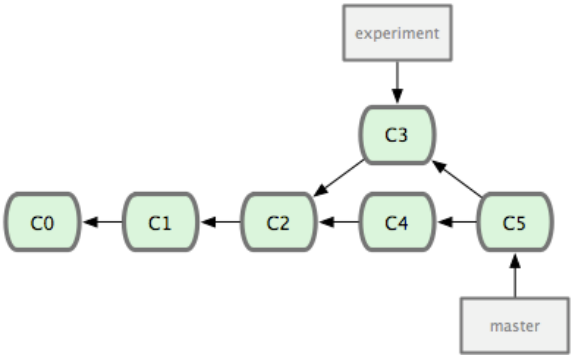
    b. Her working directory would be updated to have Version C of *cat.rb*

    c. Her working directory would be updated to have Version D of *cat.rb*

    d. The GitHub repo would be updated to have Version B of *cat.rb*

    e. None of the above

Consider this Git repo.



5. [3pts] Why does the C5 commit node point at both C3 and C4?

    a. Because C5 was created by merging data from C3 and C4

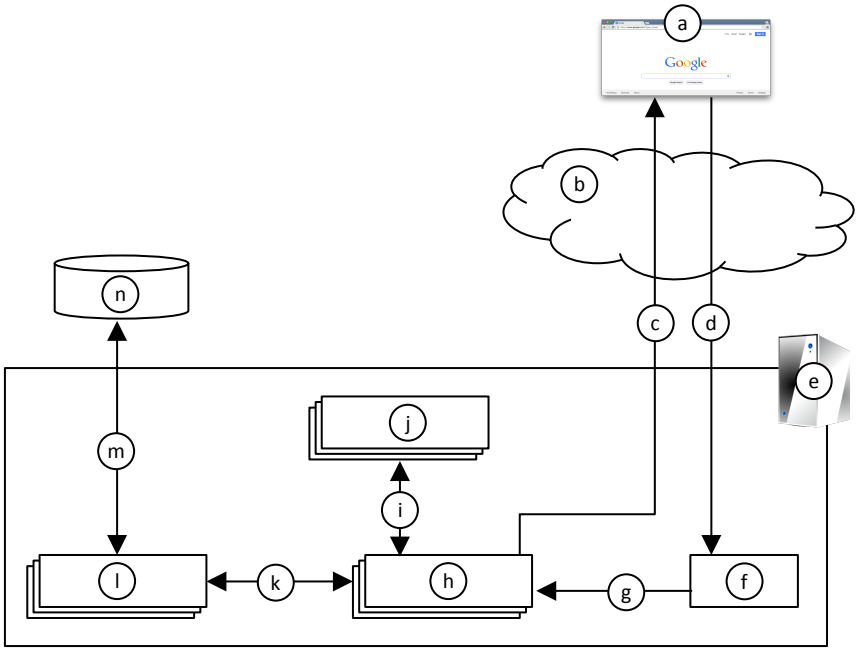
    b. Because C3 and C4 were each created by modifying data from C5

    c. This example is invalid: The node to which "experiment" directly points (C3) should not be reachable from the node to which "master" directly points (C5)

    d. This example is invalid: C5 should never point at both C3 and C4

    e. None of the above

For the next question, consider this architectural diagram:



6. [14pts] For each lettered item from the architectural diagram on the previous page, fill in the most appropriate label number.

a. _____	1) Ye Olde Internet
b. _____	2) Invocation of Model Operations
c. _____	3) Rails Controller
d. _____	4) Rendering of View
e. _____	5) SQL Queries
f. _____	6) Relational Database
g. _____	7) HTTP Response
h. _____	8) Rails Server
i. _____	9) Web Browser
j. _____	10) Rails Router
k. _____	11) Invocation of Controller Action
l. _____	12) Rails View
m. _____	13) HTTP Request
n. _____	14) Rails Model

Regarding the remaining questions:

The questions on the following pages refer to the example figures. The figures show different aspects of a WeddingHelper web app that helps a wedding planner keep track of which guests have been sent invitations and thank-you letters, and what gifts the couple received from each guest. Because each correspondence (e.g., invitation) is often sent to a household of multiple people (such as a married couple) and each gift typically comes from all the people in a household, the system organizes the guests as a set of households, each made up of one or more people.

The system has three model classes, Household, Person, and Gift (see Figure 1) and a controller class for each (not shown). Figure 2 and Figure 3 show what the index pages for households and gifts, respectively, look like. Figure 4 and Figure 5 show the ERB code for each index page (partially elided in the case of Figure 5). Figure 6 shows partially elided test code for the Person model class, and Figure 7 a form for creating a new person. (Note that Rails knows that the plural of *person* is *people*.)

7. [15pts] Draw a UML class diagram that represents the model classes given in Figure 1. Be sure to label all associations and association ends, and include all multiplicities. Don't include "id" attributes (objects have identity by default). You may also omit the datetime attributes.

8. [15pts] Write the missing ERB code in Figure 5 such that it renders pages that look like Figure 3. Do not hard code values. Rather, they should come from an `@gifts` object that was passed to the ERB. In particular, `@gifts` is an array of Gift objects.

9. [5pts] In the household index view, `@households` is an array of all the household objects. In what method was that array populated? Give the class name and method name. (These aren't shown anywhere in this exam, but you should be able to make a sensible guess.)

10. [12pts] Fill in the missing test code in Figure 6 such that the test checks that the model class' validation features will catch a missing name. Recall that all Rails model classes have a `valid?` method, and the test base class provides `assert` and `assert_not` methods.

11. [3pts] If you wanted to change the HTTP request URL that maps to a particular controller action, which Rails component would you need to modify?

    a. Controller class

    b. Model class

    c. Routes class

    d. Migration class

    e. All of the above

12. [3pts] Which of the following types of Rails components sets up the database tables?

    a. Controller classes

    b. Model classes

    c. Routes classes

    d. Migration classes

    e. All of the above

13. [3pts] What type of HTTP request would be generated by pressing the "Create Person" button in the form in Figure 7.

    a. GET

    b. POST

    c. PATCH

    d. DELETE

    e. None of the above

14. [3pts] After the HTTP request generated by Figure 7 is successfully processed on the server side, what should the server's response to the browser be?

    a. HTTP response with successful status and accompanying HTML

    b. HTTP response with unsuccessful status (404 Not Found) and no HTML

    c. HTTP redirect to another URL

    d. No response

    e. None of the above

Figures

```ruby
# Table name: households
#
#  id              :integer          not null, primary key
#  invitation_sent :boolean
#  thankyou_sent   :boolean
#  created_at      :datetime         not null
#  updated_at      :datetime         not null
#

class Household < ActiveRecord::Base
  has_many :people
  has_many :gifts
end
```

```ruby
# Table name: people
#
#  id           :integer          not null, primary key
#  name         :string
#  email        :string
#  created_at   :datetime         not null
#  updated_at   :datetime         not null
#  household_id :integer
#

class Person < ActiveRecord::Base
  belongs_to :household
  validates :name, presence: true
end
```

```ruby
# Table name: gifts
#
#  id              :integer          not null, primary key
#  name            :string
#  description     :text
#  has_receipt     :boolean
#  estimated_value :integer
#  created_at      :datetime         not null
#  updated_at      :datetime         not null
#  household_id    :integer
#

class Gift < ActiveRecord::Base
  belongs_to :household
end
```

Figure 1. Model classes for Wedding Helper web app.

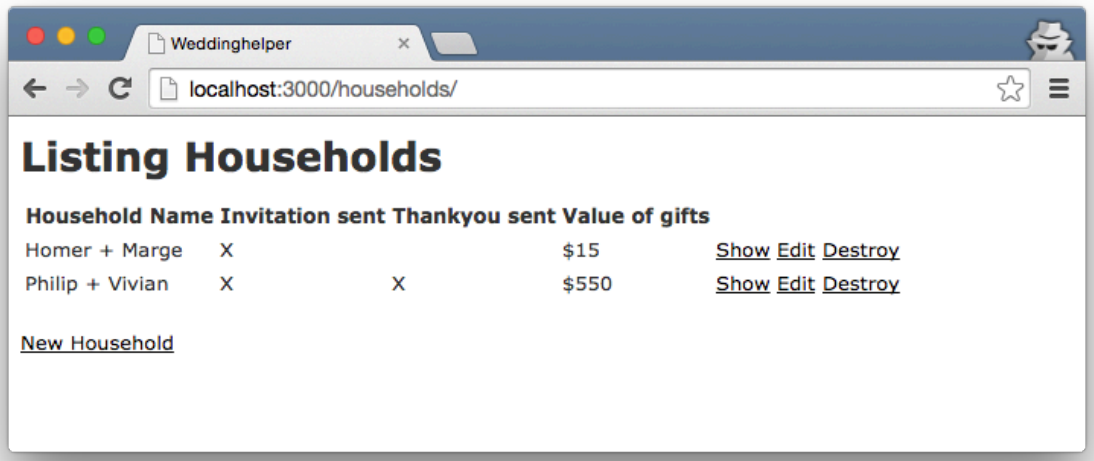


Figure 2. Index page for households.

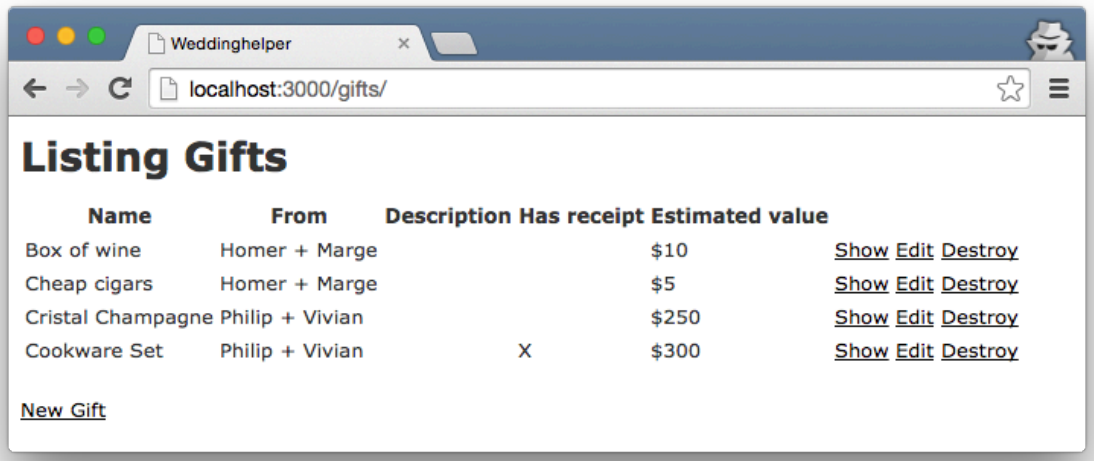


Figure 3. Index page for gifts. Note that none of the gifts has a "Description".

```
<p id="notice"><%= notice %></p>

<h1>Listing Households</h1>

<table>
  <thead>
    <tr>
      <th>Household Name</th>
      <th>Invitation sent</th>
      <th>Thankyou sent</th>
      <th>Value of gifts</th>
      <th colspan="3"></th>
    </tr>
  </thead>

  <tbody>
    <% @households.each do |household| %>
      <tr>
        <td>
          <% household.people.each do |person| %>
            <%= person.name %>
            <% if person != household.people.last %>
              +
            <% end %>
          <% end %>
        </td>
        <td><% if household.invitation_sent %>X<% end %></td>
        <td><% if household.thankyou_sent %>X<% end %></td>
        <td>
          <%
            gift_total = 0
            household.gifts.each do |gift|
              gift_total += gift.estimated_value
            end
          %>
          $<%= gift_total %>
        </td>
        <td><%= link_to 'Show', household %></td>
        <td><%= link_to 'Edit', edit_household_path(household) %></td>
        <td><%= link_to 'Destroy', household, method: :delete, data: { confirm:
          'Are you sure?' } %></td>
      </tr>
    <% end %>
  </tbody>
</table>

<br>

<%= link_to 'New Household', new_household_path %>
```

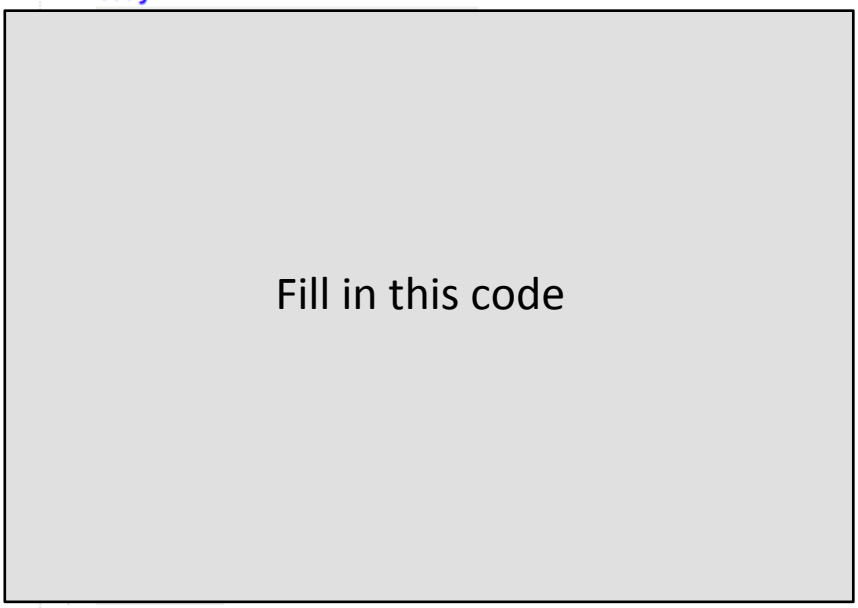
Figure 4. View code for households index page.

```
<p id="notice"><%= notice %></p>

<h1>Listing Gifts</h1>

<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>From</th>
      <th>Description</th>
      <th>Has receipt</th>
      <th>Estimated value</th>
      <th colspan="3"></th>
    </tr>
  </thead>

  <tbody>
```




```
  </tbody>
</table>

<br>

<%= link_to 'New Gift', new_gift_path %>
```

Figure 5. Partially elided view code for gifts index page.

```ruby
require 'test_helper'

class PersonTest < ActiveSupport::TestCase

  def setup
    @person = Person.new(name: "Homer", email: "homer@example.com")
  end

  test "name should be present" do

Fill in this code

  end

end
```

Figure 6. Model test case with elided code.

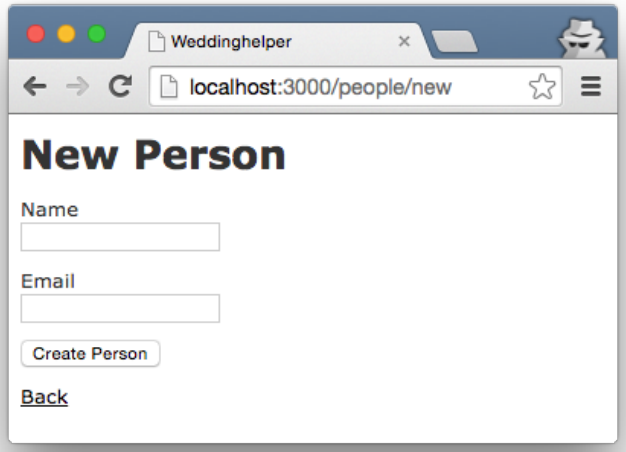


Figure 7. Form for creating a new person.