COMP 4081

Exam 1

Fall 2014

Name: Solutions ,
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. [3pts] The Git **clone** command does which of the following?

   a. Creates a working directory

   b. Makes a local copy of the repository

   c. Commits a new branch

   (d.) a and b

   e. a, b, and c

2. [3pts] Which Git command changes where the HEAD pointer points and modifies the contents of the working directory?

   (a.) checkout

   b. merge

   c. mv

   d. pull

   e. None of the above

3. [3pts] Which one of the following is <u>not</u> part of the data structure of a Git repository?

   (a.) Body element

   b. Branch pointer

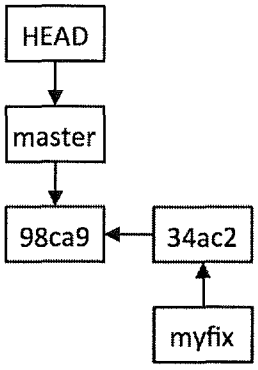
   c. Commit object

   d. HEAD pointer

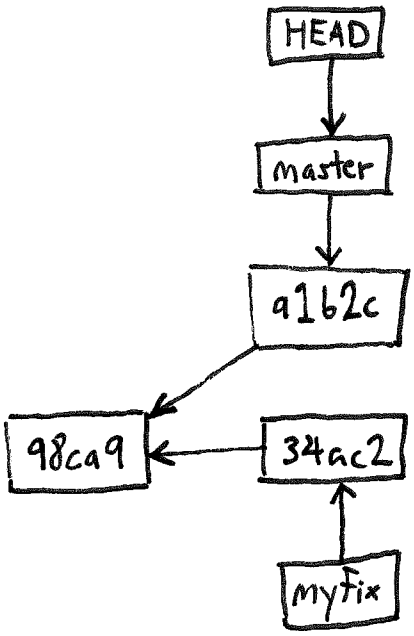
   e. None of the above (i.e., they are all parts)

4. [8pts] Draw the state of the pictured repository after a Git **commit** operation (make up a hash).

Before:



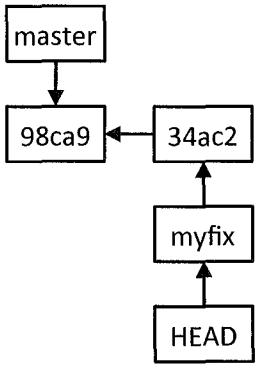
After:



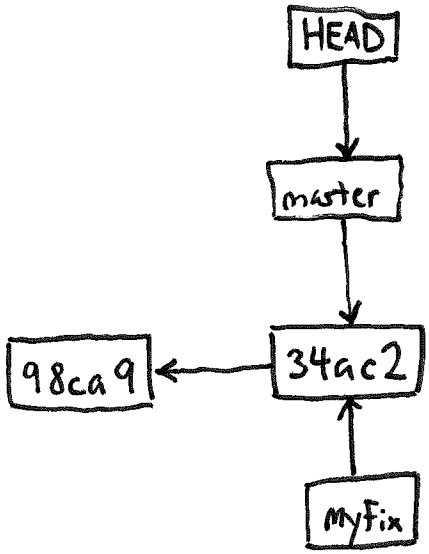
5. [8pts] Draw the state of the pictured repository after running the following commands.

```
$ git checkout master
$ git merge myfix
```

Before:



After:



6. [8pts] Given the Rails MVC architectural diagram below, label each component.

a. Web Browser

b. Ye Olde Internet

c. Rails Router

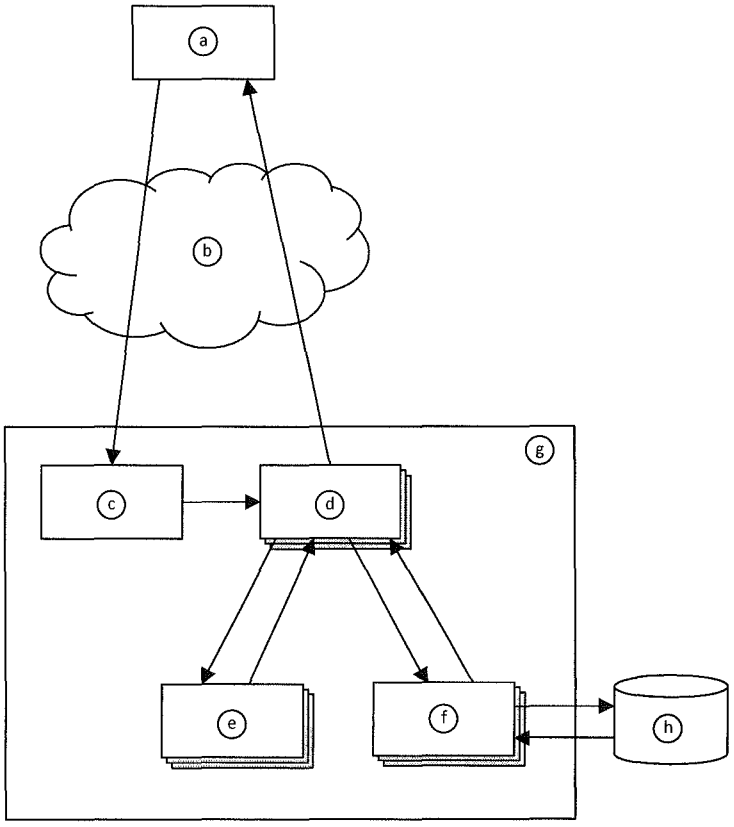
d. Controller

e. View

f. Model

g. Rails Server

h. Database



7. [14pts] Figures 1–6 pertain to a rental-property web app. Write Ruby code that defines the `show` method in Figure 5, and write the ERB code that would produce the page depicted in Figure 2. Assume that a layout, application.html.erb, already exists, so your ERB needs only to include the main content being displayed. Your ERB must have the following types of HTML elements: **p** and **strong**.

```
def show
    @rental = Rental.find(params[:id])
end
```

```
<p>
  <strong>Address:</strong> <%= @rental.address %>
</p>
<p>
  <strong>Rent:</strong> <%= @rental.rent %>
</p>
<p>
  <strong>Bedrooms:</strong> <%= @rental.bedrooms %>
</p>
<p>
  <strong>Bathrooms:</strong> <%= @rental.bathrooms %>
</p>
<p>
  <strong>Landlord:</strong> <%= @rental.landlord %>
</p>
<p>
  <strong>Phone:</strong> <%= @rental.phone %>
</p>
```

cont'd next page

```
<%= link_to 'Edit', edit_rental_path(@rental) %>
<%= link_to 'Back', rentals_path %>
```

8. [3pts] Why would it violate the SRP to move line 3 from `RentalsController` (Figure 5) into the beginning of index.html.erb (Figure 6)?

It would violate the single-responsibility principle (SRP) because a controller is responsible for translating between UI actions and operations on the model, whereas a view is responsible for UI presentation. Line 3 is an operation on the model — a controller responsibility. Moving this line into the view would mean that the view now has both view and controller responsibilities.

9. [3pts] Which of the following is true of *exhaustive testing*?

   a. Generally infeasible in practice

   b. Tests all possible inputs

   c. Typically results in an intractably large set of test cases even for small programs

   (d.) All of the above

   e. None of the above

10. [3pts] Which type(s) of tests do you typically write when doing test-driven development?

   a. Automated tests

   b. Unit tests

   c. Blackbox tests

   (d.) All of the above

   e. None of the above

11. [9pts] Following test-driven development (TDD), describe in plain English how you would go about adding a new method `index2br` to `RentalsController` along with an associated ERB. The purpose of this functionality would be to display an index of all rentals with 2 bedrooms. Describe the sequence of steps you would perform and all the tests you would write. Cover all aspects of TDD.

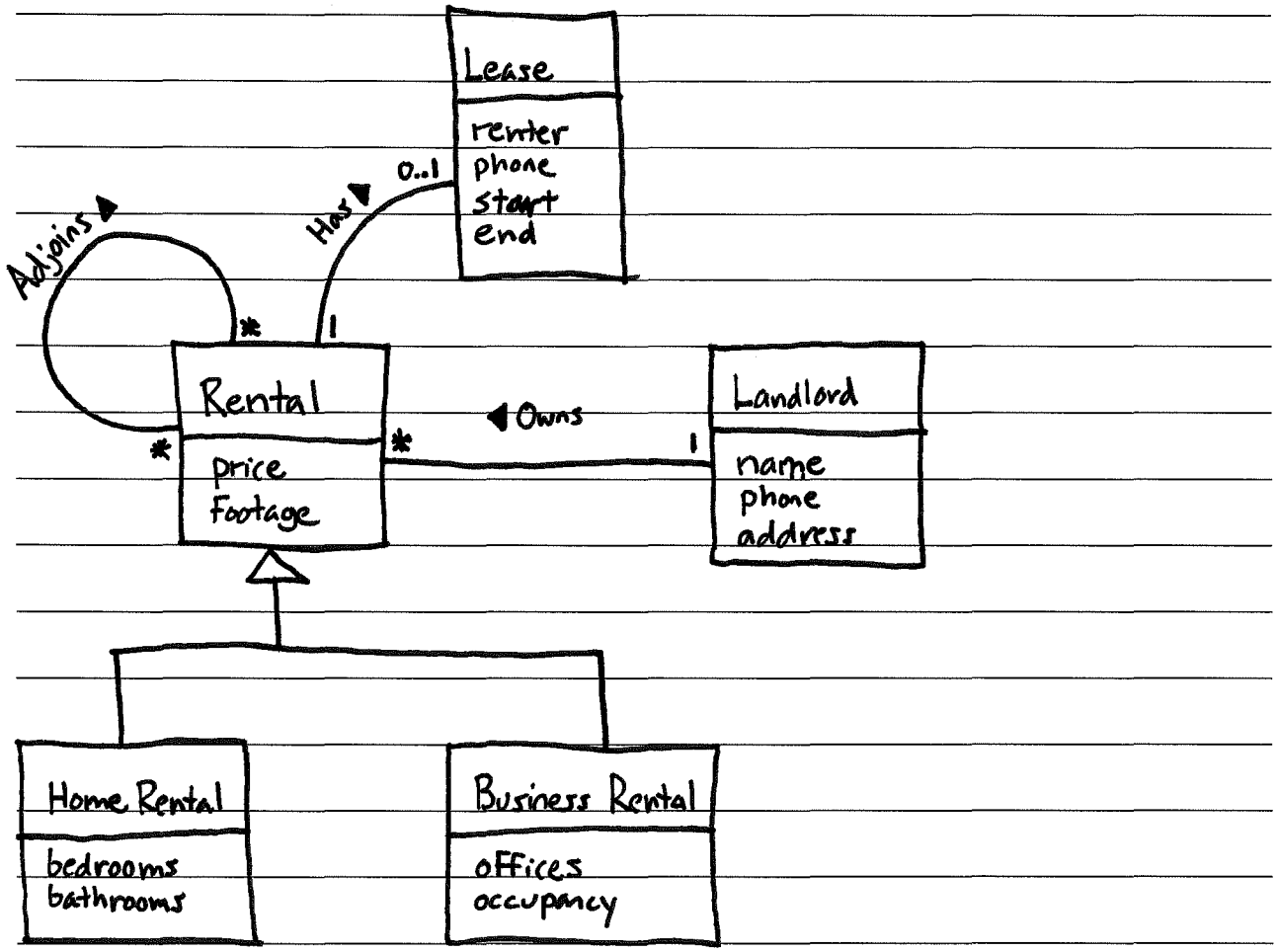
Red: Following TDD, I would first write tests for the new functionality. Since these would be black box tests, I would have one or more "normal" cases (e.g., a mix of 2-bedroom and non-2-bedroom rentals). I would also have some boundary cases: no rentals at all, no 2-bedroom rentals, only 2-bedroom rentals, etc. Having written the tests, I would run them to see that they fail.

Green: I would then write just enough code to make the test pass. (the index2br method and ERB)

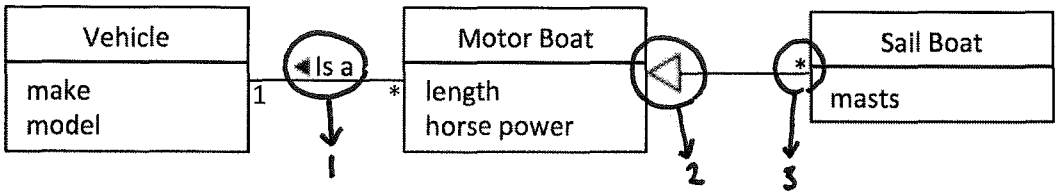
Refactor: Finally, if I saw an opportunity to improve my design after implementing the code, I would refactor the code, improving the design.

12. [17pts] Draw a domain model in the form of a class diagram based on the following description. Model only things that are specifically described. Include all conceptual classes, attributes, associations, and generalization relationships mentioned. Label all associations and include all multiplicities. Do not model "the system."

A system is needed for managing rental properties. Each rental property has a monthly rent price and square footage. Each rental is owned by a landlord with a name, a phone number, and an address. There are two types of rentals: home and business. Home rentals have a number of bedrooms and a number of bathrooms. Business rentals have a number of offices and a maximum occupancy. A rental may adjoin a number of other rental properties (like apartments in the same complex). If a rental property is currently rented, it will have an associated lease with the name of the renter, the renter's phone number, a start date, and an end date.



13. [3pts] Name three problems with this domain model (esp. with respect to class diagram notation).



1. An "is-a" relationship should be modeled with a generalization (i.e., an arrow like this: ◁—).
2. This generalization violates both the Is-a and 100% rules: Not all sail boats have motors (with horse power).
3. Generalization relationships should not have multiplicities.

14. [3pts] Which of the following is meant by *software engineering process*?

(a.) A structure imposed on the development of a software product; for example, developing iteratively and incorporating best practices might be ingredients in an SE process

b. Something developers use to accomplish a goal during a project; for example, Git is an SE process for configuration management

c. Something developers do to accomplish a goal during a project; for example, planning poker is an SE process for estimation

d. A running instance of a program; for example, a UNIX process is an SE process

e. None of the above

15. [3pts] An empirical process-control model iterates between _____.

a. design and implementation

b. requirements gathering and design

c. user studies and testing

(d.) feedback and adaptation

e. None of the above

16. [9pts] Please answer the following 3 related questions:
    a. What often-false assumption does the waterfall software engineering process make?
    b. Why does this false assumption cause considerable problems for waterfall?
    c. How does iterative development overcome these problems?

(a) WaterFall assumes that requirements are stable (low rate of change) and can be known from the start.

(b) This assumption causes considerable problems because of the Defect Cost Increase (DCI) Principle — the later you discover a defect, the more expensive to fix. In waterfall, testing and customer feedback come very late in the process; thus, defects are often discovered late.

(c) Iterative development addresses this problem with short iterations (2-4 weeks). By the end of each iteration tests are run and feedback collected; thus, issues with the requirements are discovered quickly.

Figures

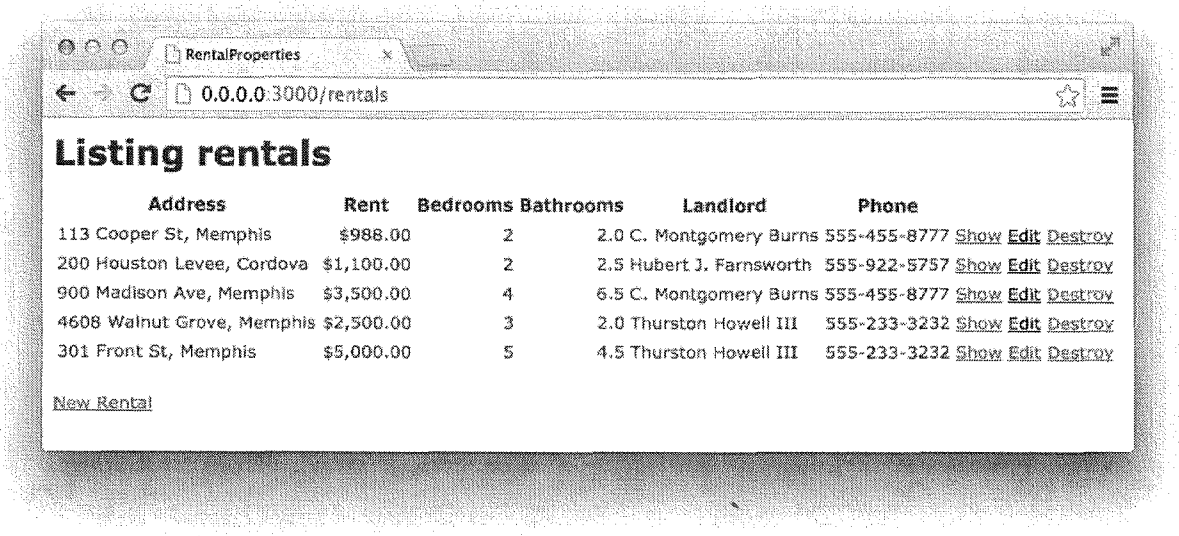


Figure 1. Index page for rental-property web app.

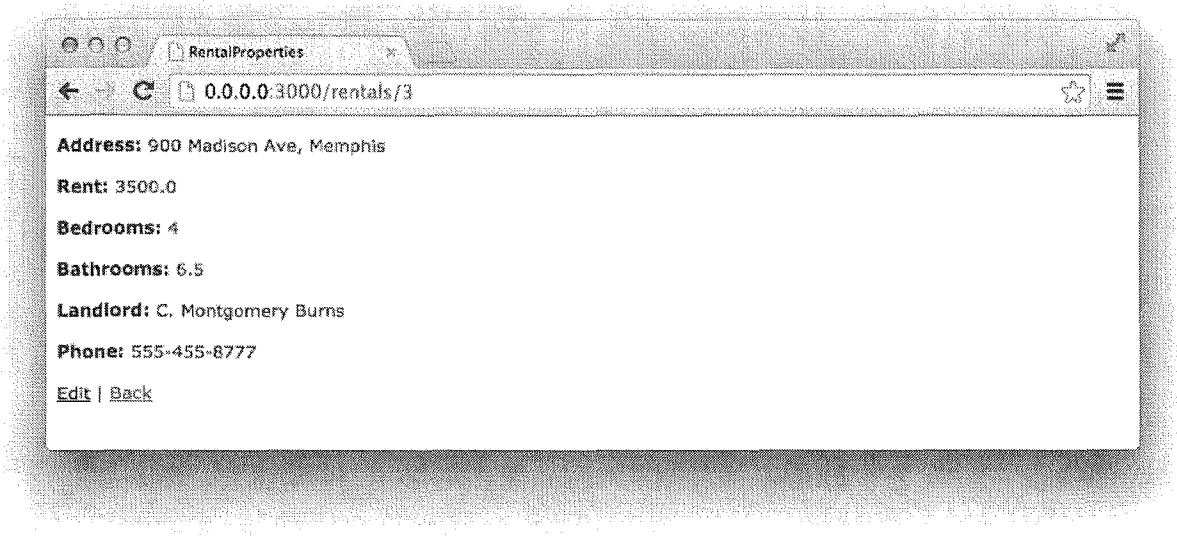


Figure 2. Show-rental page for rental-property web app.

```
$ rake routes
     Prefix Verb   URI Pattern                  Controller#Action
    rentals GET    /rentals(.:format)           rentals#index
            POST   /rentals(.:format)           rentals#create
 new_rental GET    /rentals/new(.:format)       rentals#new
edit_rental GET    /rentals/:id/edit(.:format)  rentals#edit
     rental GET    /rentals/:id(.:format)       rentals#show
            PATCH  /rentals/:id(.:format)       rentals#update
            PUT    /rentals/:id(.:format)       rentals#update
            DELETE /rentals/:id(.:format)       rentals#destroy
```

Figure 3. Result of "rake routes" command for rental-property web app.

```ruby
# == Schema Information
#
# Table name: rentals
#
#  id          :integer          not null, primary key
#  address     :string(255)
#  rent        :decimal(, )
#  bedrooms    :integer
#  bathrooms   :float
#  landlord    :string(255)
#  phone       :string(255)
#  created_at  :datetime
#  updated_at  :datetime
#

class Rental < ActiveRecord::Base
end
```

Figure 4. Rental-property web app file: app/models/rental.rb

```ruby
class RentalsController < ApplicationController
  def index
    @rentals = Rental.all
  end

  def show
    # YOUR ANSWER HERE
  end

  def new
    @rental = Rental.new
  end

  def edit
    @rental = Rental.find(params[:id])
  end

  def create
    @rental = Rental.new(rental_params)
    respond_to do |format|
      if @rental.save
        format.html { redirect_to @rental, notice: 'Rental was successfully created.' }
        format.json { render action: 'show', status: :created, location: @rental }
      else
        format.html { render action: 'new' }
        format.json { render json: @rental.errors, status: :unprocessable_entity }
      end
    end
  end
```

... and so on ...

Figure 5. Rental-property web app file: app/controllers/rentals_controller.rb

```
<h1>Listing rentals</h1>

<table>
  <thead>
    <tr>
      <th>Address</th>
      <th>Rent</th>
      <th>Bedrooms</th>
      <th>Bathrooms</th>
      <th>Landlord</th>
      <th>Phone</th>
      <th></th>
      <th></th>
      <th></th>
    </tr>
  </thead>

  <tbody>
    <% @rentals.each do |rental| %>
      <tr>
        <td><%= rental.address %></td>
        <td style="text-align: right;"><%= number_to_currency(rental.rent) %></td>
        <td style="text-align: right;"><%= rental.bedrooms %></td>
        <td style="text-align: right;"><%= rental.bathrooms %></td>
        <td><%= rental.landlord %></td>
        <td><%= rental.phone %></td>
        <td><%= link_to 'Show', rental %></td>
        <td><%= link_to 'Edit', edit_rental_path(rental) %></td>
        <td><%= link_to 'Destroy', rental, method: :delete, data: { confirm: 'Are you sure?' } %>
        </td>
      </tr>
    <% end %>
  </tbody>
</table>

<br>

<%= link_to 'New Rental', new_rental_path %>
```

Figure 6. Rental-property web app file: app/views/index.html.erb