COMP 4081

Exam 1

Fall 2018

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. [2pts] What type of system is Git?

Distributed version control system

2. [2pts] What type of system is GitHub?

Web-based Git repository hosting service

3. [2pts] Which of the following problems does Git help solve?

   a) Developing multiple versions of a software project in parallel

   b) Merging changes to a software project made by collaborating developers working in parallel

   c) Recovering older versions of a software project

   (d) All of the above

   e) None of the above

4. [2pts] Which of the following does <u>not</u> help with Rails development environment configuration management?

   a) Bundler

   (b) GitHub

   c) RVM

   d) Vagrant

   e) None of the above (i.e., all help with configuration management)

↑ Also acceptable because an argument can be made...

Consider the following list of Git commands:

a)	`git status`	f)	`git branch`
b)	`git pull`	g)	`git commit`
c)	`git merge`	h)	`git clone`
d)	`git checkout`	i)	`git add`
e)	`git push`	j)	`git init`

Keven has just joined a team of developers collaboratively working on a skating competition results web app called *MeetManager*. The code for the project is housed in a GitHub repo. All work for the project is being done on the "master" branch (no other branches).

5. [2pts] Keven wants to get a local copy of the code and repo, so he can begin contributing to the project. Which command(s) from the above list should he run?

   h

6. [2pts] He makes some changes to the code. Which command(s) from the above list should he run to save his edits to the local repo?

   i, g

7. [2pts] Having saved to his local repo, he would now like to share his work with the rest of the team (via GitHub). Which command(s) from the above list should he run?

   e

When he runs the command(s), he gets this message (with words that give away the answers hidden):

```
To https://github.com/.../meetmanager.git
 ! [rejected]        master -> master (fetch first)
error: failed to [redacted] some refs to 'https://github.com/.../meetmanager.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository [redacted]ing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git [redacted] ...') before [redacted]ing again.
hint: See the 'Note about fast-forwards' in 'git [redacted] --help' for details.
```

8. [2pts] Keven wants to resolve this issue, so he can upload her changes. Which (one) command from the above list should he run next?

   b

9. [2pts] The command completes with no conflicts. What command(s) should Keven run to, at last, share his work with the rest of the team?

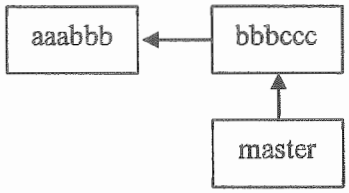
   e

10. [2pts] If Keven wanted to look at a previous version of the code, which command would he use?

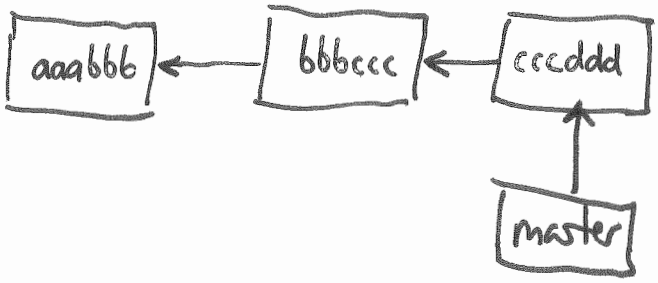
    d

11. [4pts] Draw the state of the pictured repository after a Git **commit** operation (call your hash *cccddd*).

Before:

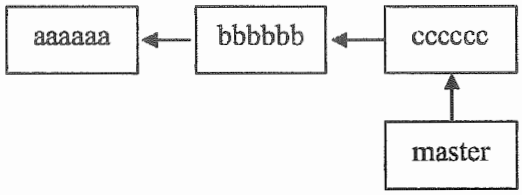


After:

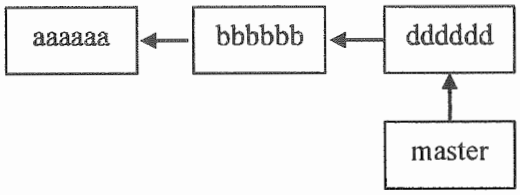


12. [4pts] Draw the state of the local repository after a Git **pull**. Assume that auto-merge runs successfully (with no conflicts). If creating any new commits, follow the naming convention in the diagram.

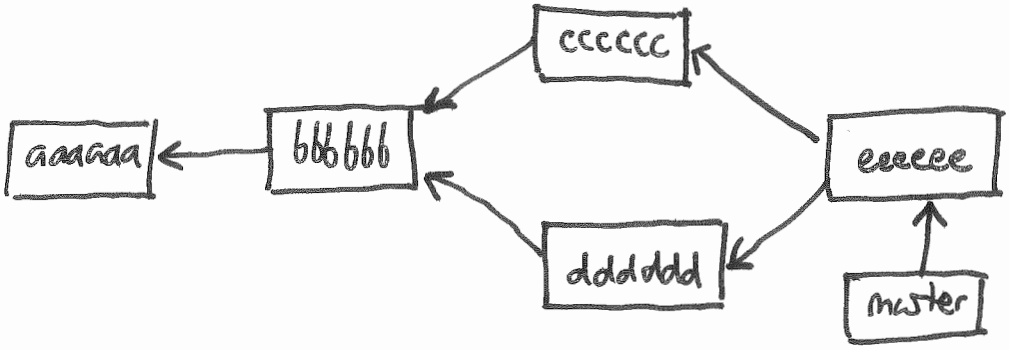
Remote Repo:



Local Repo Before:

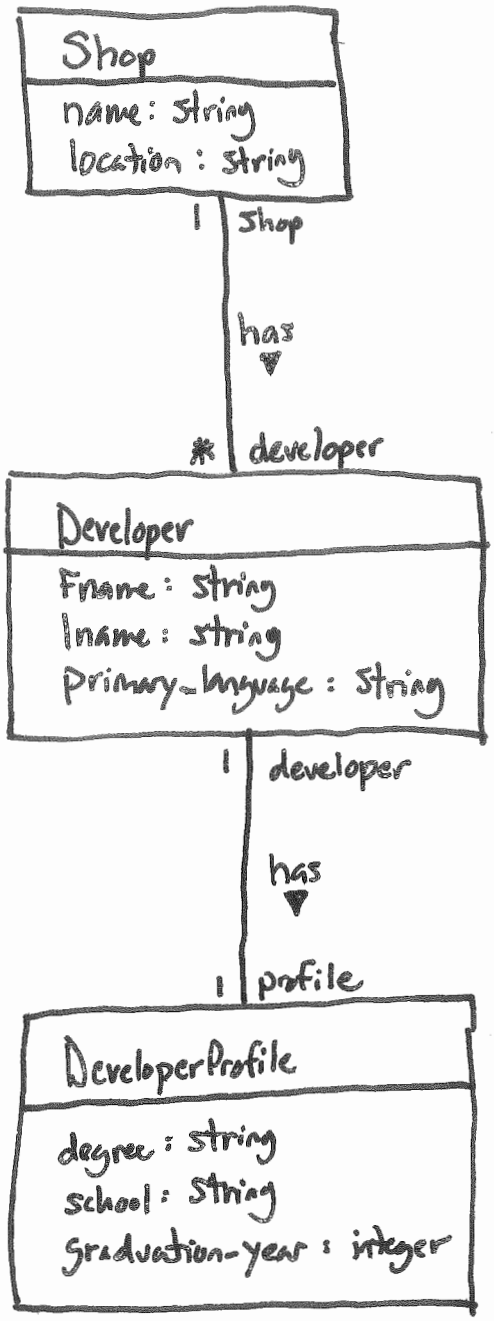


Local Repo After:



The questions on the following pages refer to the example figures. The figures show different aspects of the *CodeGuru* web app that helps an individual or company find a suitable software shop or programmer for their project. Users can use the app to browse software shops and view a shop's developers and their backgrounds.

13. [10pts] Draw a UML class diagram that represents the three model classes given in Figure 1. Be sure to label all associations and association ends and include all multiplicities. Don't include any "id" attributes (including foreign keys). You may also omit the "datetime" attributes that Rails provides by default.



14. [6pts] Consider the model classes in Figure 1 and the fixtures in Figure 2. Using the lines of code in Figure 3, complete the following model test classes such that each model class has test for a valid instance of the class and such that each validation has a test which demonstrates that the validation catches an invalid value. You should fill all blanks and use all lines at least once. Some lines may be used more than once.

```
class ShopTest < ActiveSupport::TestCase
  M
  F
  L
  K
  E
  F
  I
  C
  K
end
```

```
class DeveloperTest < ActiveSupport::TestCase
  A
  G
  L
  K
  H
  G
  O
  C
  K
end
```

(Continued next page…)

```
class DeveloperProfileTest < ActiveSupport::TestCase
  J
  B
  L
  K
  D
  B
  N
  C
  K
end
```

15. [9pts] Consider the Developers *index* page in Figure 4. Using the lines of code in Figure 6, reverse engineer the view code that produced this page. You should fill every blank and use all lines at least once. Some lines may be used more than once.

L
B
I
G
F
A
U
K
C
H
D
Q
G
V
R
O
T
S
J
C
M
N
E
P

16. [3pts] By inserting two lines of code, it is possible to add a "Shop" column to the Developers *index* page that lists the associated shop name for each developer. What are the two lines of code, and where should they be inserted in your answer to the previous question?

Insert after line (u) <th> Primary language </th> :
<th> Shop name </th>

Insert after line (o) <td><%= developer.primary_language %></td>
<td><%= developer.shop.name %></td>

17. [2pts] Which of the following routes is used to display the form in Figure 5?

a) `get '/developers/:id', to: 'developers#show', as: 'developer'`

b) `patch '/developers/:id', to: 'developers#update'`

c) `post '/developer', to: 'developers#create'`

(d) `get '/developers/:id/edit', to: 'developers#edit', as: 'edit_developer'`

e) `get '/developers', to: 'developers#index', as: 'developers'`

18. [2pts] Which of the following lines of code would the controller need to execute before rendering the form view from Figure 5?

a) `@developers = Developer.all`

b) `@developer = Developer.new`

(c) `@developer = Developer.find(params[:id])`

d) `@developer = Developer.new(params.require(:developer).permit(:fname,:lname,:primary_language,:shop_id))`

e) None of the above

19. [1pt] True or false? State-affecting controller actions (such as create, update, and destroy) should always render a view, which produces an HTTP response containing HTML for the browser to display.

a) True

(b) False

20. [9pts] For each component below, give the corresponding letter from the Rails architectural diagram in Figure 7.

G Model

A Browser

E Model Tests

H Controller

B Migrations

C Internet

D Database

F View

I Router

Figures

```
# == Schema Information
#
# Table name: shops
#
#  id            :integer          not null, primary key
#  name          :string
#  location      :string
#  created_at    :datetime         not null
#  updated_at    :datetime         not null
#

class Shop < ApplicationRecord
    has_many :developers
    validates :name, presence: true
end
```

```
# == Schema Information
#
# Table name: developers
#
#  id                   :integer          not null, primary key
#  fname                :string
#  lname                :string
#  primary_language     :string
#  shop_id              :integer
#  created_at           :datetime         not null
#  updated_at           :datetime         not null
#
# Indexes
#
#  index_developers_on_shop_id  (shop_id)
#

class Developer < ApplicationRecord
  belongs_to :shop
  has_one :developer_profile
  validates :primary_language, inclusion: { in: ['Java', 'Python', 'C#', 'Ruby', 'PHP'] }
end
```

```
# == Schema Information
#
# Table name: developer_profiles
#
#  id                  :integer          not null, primary key
#  degree              :string
#  school              :string
#  graduation_year     :integer
#  developer_id        :integer
#  created_at          :datetime         not null
#  updated_at          :datetime         not null
#
# Indexes
#
#  index_developer_profiles_on_developer_id  (developer_id)
#

class DeveloperProfile < ApplicationRecord
  belongs_to :developer
  validates :graduation_year, numericality: { only_integer: true, less_than_or_equal_to:
    Date.current.year }
end
```

Figure 1. Three model classes from the CodeGuru app.

```
one:
  name: Helium
  location: Atlanta, GA

two:
  name: Northwest Independent Ruby Development
  location: Seattle, WA
```

```
one:
  fname: John
  lname: Harrington
  primary_language: Java
  shop: one

two:
  fname: Mary
  lname: Baldwin
  primary_language: Ruby
  shop: two
```

```
one:
  degree: Masters, Computer Science
  school: University of Chicago
  graduation_year: 2008
  developer: one

two:
  degree: Bachelors, Computer Science
  school: University of Memphis
  graduation_year: 2016
  developer: two
```

Figure 2. Test fixtures for the CodeGuru model classes.

```
(a) test "should be valid developer" do
(b) one = developer_profiles(:one)
(c) assert_not one.valid?
(d) test "should be invalid developer_profile" do
(e) test "should be invalid shop" do
(f) one = shops(:one)
(g) one = developers(:one)
(h) test "should be invalid developer" do
(i) one.name = ''
(j) test "should be valid developer_profile" do
(k) end
(l) assert one.valid?
(m) test "should be valid shop" do
(n) one.graduation_year = 2025
(o) one.primary_language = 'Perl'
```

Figure 3. Model unit test lines of code.

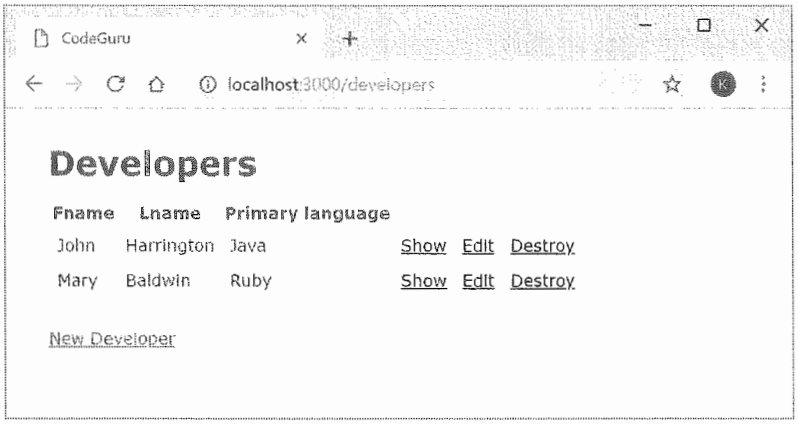


Figure 4. Developers *index* page.

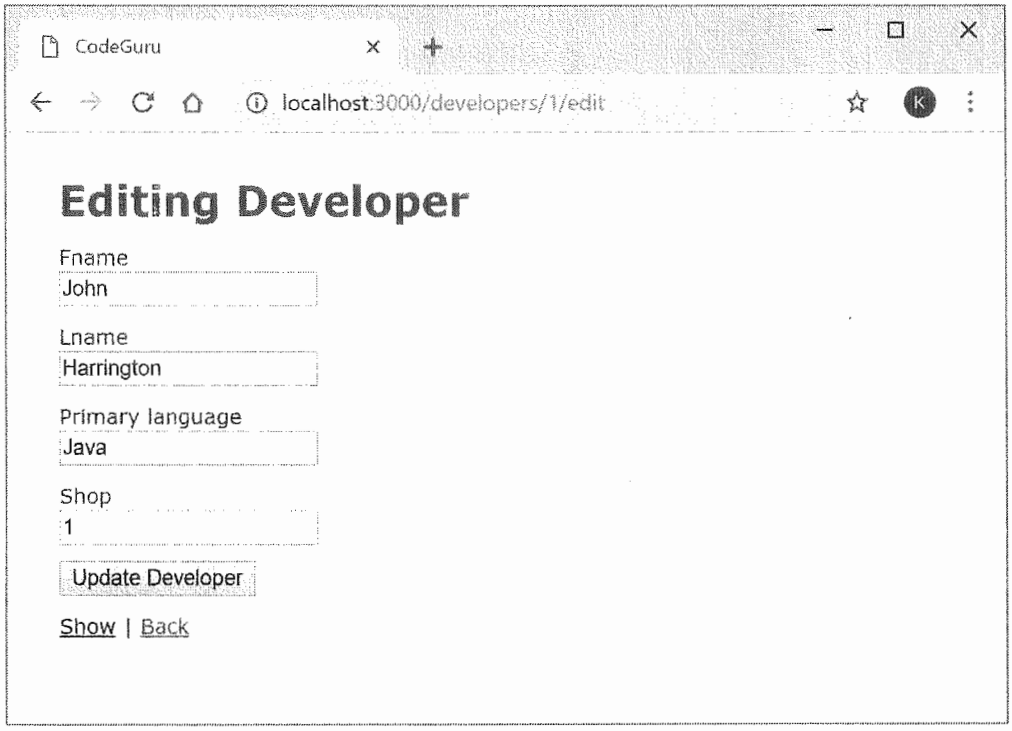


Figure 5. Form for updating a Developer.

```
(a) <th>Lname</th>
(b) <table>
(c) </tr>
(d) <tbody>
(e) </table>
(f) <th>Fname</th>
(g) <tr>
(h) </thead>
(i) <thead>
(j) <td><%= link_to 'Destroy', developer, method: :delete, data: { confirm: 'Are you
    sure?' } %></td>
(k) <th colspan="3"></th>
(l) <h1>Developers</h1>
(m) <% end %>
(n) </tbody>
(o) <td><%= developer.primary_language %></td>
(p) <%= link_to 'New Developer', new_developer_path %>
(q) <% @developers.each do |developer| %>
(r) <td><%= developer.lname %></td>
(s) <td><%= link_to 'Edit', edit_developer_path(developer) %></td>
(t) <td><%= link_to 'Show', developer %></td>
(u) <th>Primary language</th>
(v) <td><%= developer.fname %></td>
```

Figure 6. Lines of ERB code for the Developers *index* page.

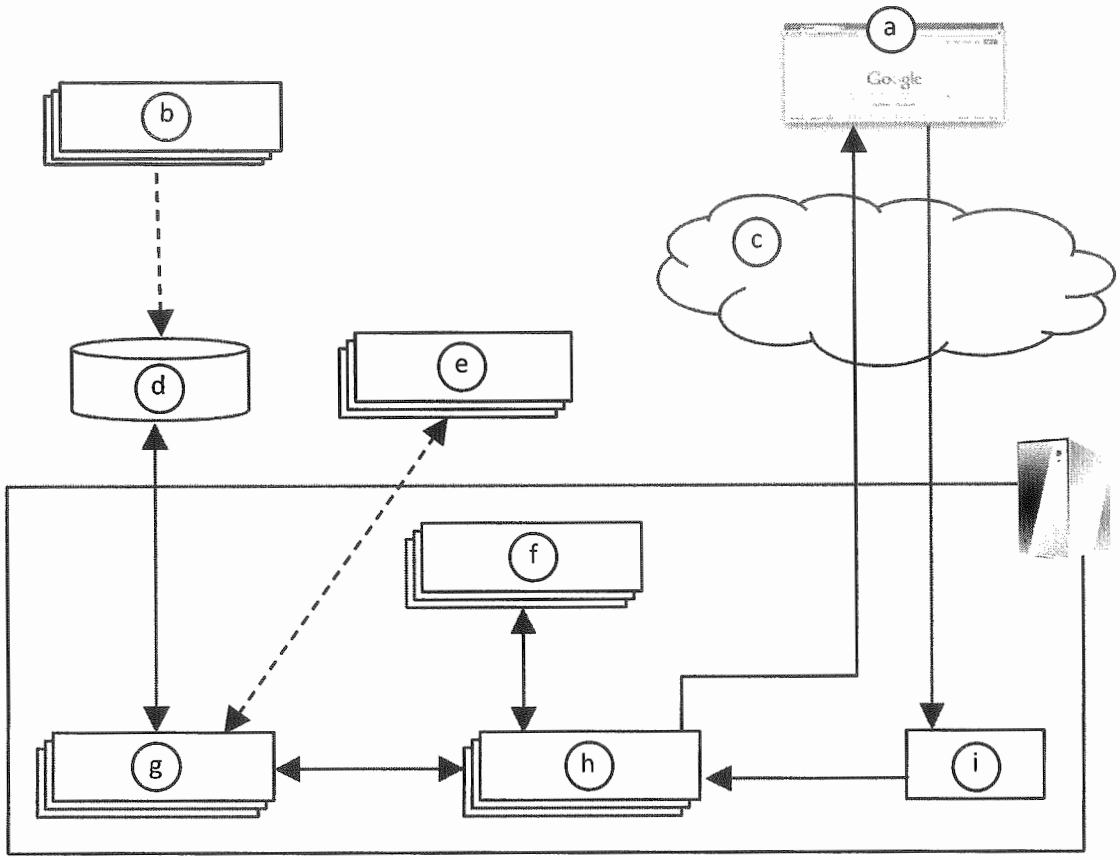


Figure 7. Rails architectural diagram.