Name ____________________________

Midterm Exam for IT 232
Spring 2013

For multiple choice (1 point each), circle the best answer.

1. A model attribute that refers to an instance in another model is called
   a) a primary key.
   b) a foreign key.
   c) a universal key.
   d) an id key.

2. For checking valid user input when saving to the database, Rails validation code should be added to
   a) the routing system.
   b) the test folder.
   c) the controller.
   d) the model.

3. The Rails component that parses the URL to determine the controller and action is
   a) the model.
   b) the view.
   c) the database.
   d) routing system.

4. To implement a many-to-many relationship between two Rails models,
   a) the belongs_to and has_many methods are needed.
   b) a join table is used.
   c) the belongs_to and has_one methods are needed.
   d) foreign keys are added to each model.

5. Assume that **m** is a reference to a Movie object. In the Ruby expression **m.save**, **save** is
   a) an instance method.
   b) a class method.
   c) a virtual attribute.
   d) an accessible attribute.

6. Assume that **Movie** is the name of a data model. In the Ruby expression **Movie.find_by_title("Jaws")**, **find_by_title** is
   a) an instance method.
   b) a class method.
   c) a virtual attribute.
   d) an accessible attribute.

7. The following is a list of events that occur when a simple search form is submitted to a Rails application that searches the database for records that match its search phrase. The events are not listed in order. Indicate the correct order by placing numbers in the blanks before each event. (the first and last events are already numbered for you) (4 points)

   _7_ The rendered page with matching records is returned to the web browser.

   _4_ The controller obtains the parameter values from the form to prepare the database query.

   _3_ The routing system calls the appropriate controller action.

   _6_ The view file formats the matching records for display.

   _1_ The user completes the form and clicks on the form's submit button.

   _2_ The routing system parses the requesting URL.

   _5_ The controller queries the database for matching records.

8. The Rails scaffold uses a model-based form. Present one way that a model-based form is different than a simple form in Rails. (2 points)

   Possible answers include:

   - Model-based form requires an object and uses its attributes to fill out the form with values; the simple form doesn't involve an object.
   - The model-based form automatically highlights where validation errors occur; the simple form does not support validation issues.
   - The simple form stores parameter values at the top level in the params table. The model-based form stores parameter values at a second level (under params[:model_name]).
   - The simple form uses different helper methods (e.g. form_tag, select_tag) than the model-based form (e.g. form_for, select).

For the remaining questions, assume the model schema and contents listed below. These are the same models and records presented in the practice quiz.

- The airports table has the following records:

id	name	code	flights	city_id
1	O'Hare	ORD	882000	1
2	Midway	MDW	250000	1
3	Benito Juárez	MEX	378000	4
4	Heathrow	LHR	475000	2
5	JFK International	JFK	409000	3
6	LaGuardia	LGA	362000	3

- The cities table has the following records:

id	name	population	country
1	Chicago	3000000	USA
2	London	8000000	UK
3	New York	8000000	USA
4	Mexico City	8800000	Mexico

- The Airport and City models have been declared so that an airport belongs to a city and that a city has many airports.

- The Airport model has the following virtual attribute defined in its model file:

```
def long_name
   city.name + " " + name + " (" + code + ")"
end
```

9. Provide the values of the following Ruby expressions (1 point each):

a) `Airport.find(2).code`

`MDW`

b) `Airport.find_by_code('MDW').name`

`Midway`

c) `City.find(1).airports.first.code`

`ORD`

d) `Airport.find(1).long_name`

`Chicago O’Hare (ORD)`

e)
```
limit = 400000
Airport.where('flights > ?', limit).count
```

`3`

10. Provide Ruby code that performs the specified results. The code should work for any database content (not just what is listed in the last page). The first answer is provided as an example. (2 points each)

a. Code that provides the name of the city whose id is 2.

```
City.find(2).name
```

b. Code that provides the number of flights for the “Heathrow” airport.

```
City.find_by_name('Heathrow').flights
```

c. Code that lists (as an array) the city objects with population greater than 4000000.

```
City.where('population > 4000000')
```

d. Code that provides the number of airports for New York city.

```
City.find_by_name('New York').airports.count
```

11. Below show how the following view code would appear in a browser (4 points):

```
<%
  choice = City.find_by_name("Chicago")
%>

<h1>Options for <%= choice.name %></h1>


<% choice.airports.each do |item| %>

<p>
<%= item.code %> <br />
<%= item.flights %>
</p>

<% end %>


<p>Count for <%= choice.name %>: <%= choice.airports.count %>
</p>
```

Present how the page would appear here:

Options for Chicago

ORD
882000

MDW
250000

Count for Chicago: 2

12. The following form allows the user to select an airport and a city so that the chosen airport will be assigned to the selected city.

```
<%= form_tag assign_airports_path do %>

<p>
<%= label_tag :city_name %>
<%= select_tag :city_name,
    options_for_select(City.all.collect { |city| city.name },
params[:city_name] ) %>
</p>

<p>
<%= label_tag :airport_code %>
<%= select_tag :airport_code,
    options_for_select(Airport.all.collect { |a| a.code },
params[:airport_code] ) %>
</p>

<p>
<%= submit_tag 'Assign airport' %>
</p>

<% end %>
```

This form submits the city as its name (params[:city_name]) and the airport as its code (params[:airport_code]). Below, write the code for the controller action so that the specified airport object is assigned to the selected airport. Your controller code will need to reference the given parameter values. (3 points)

```
def assign
   # insert your code here
   city_obj = City.find_by_name(params[:city_name])
   airport_obj = Airport.find_by_code(params[:airport_code])

   city_obj.airports << airport_obj
end
```

Alternative method:

```
def assign
   # insert your code here
   city_obj = City.find_by_name(params[:city_name])
   airport_obj = Airport.find_by_code(params[:airport_code])

   airport_obj.city_id = city_obj.id
   airport_obj.save
end
```