Lab 3 Check-Off

Student Name: _____

Date of Lab: _____

Check-Offs

_____ The temperature conversions method is properly constructed and functioning correctly.

_____ Git repository created and git log show appropriate history.

_____ The time functionality is completed and all answers are correct.

TA (signed): _____

TA (print): _____

Lab 3 Instructions

Lab Objectives: The primary objective of this lab to help students learn the basics of Ruby programming. Secondary objectives include using basic git commands, learning how to work with the Time class in Ruby and using `irb` command prompt.

Directions:

0. Install Rails 3.2.1: Before going further, we are going to upgrade the course version of Rails to 3.2.1. Assuming you have Ruby set up properly from week 1, this should be a pretty easy process. On the command line type `gem update rails --no-ri --no-rdoc` and the install process will begin. (In some cases you may have to prefix this command with `sudo`.) Once completed, type gem list and verify that the rails gem is now at 3.2.1.

1. We are going to begin by writing a simple program for Ruby to convert temperatures and then modify it several times. To begin, create a new file called temp_conversions.rb using your preferred editor/IDE.

2. In this file, create a method called convert as follows:

```
def convert(temp)

end
```

   Between these two lines, add in the following line of code to convert Fahrenheit to Celsius:

```
5 * (temp - 32)/9
```

   After the end statement, create some simple tests for this method with the code below:

```
puts convert(32)
puts convert(50)
puts convert(212)
```

Run this code from the command line by changing to the same directory as the file and typing `ruby temp_conversions.rb`. You should get the output 0, 10, 100 as a result. Once this is running, set up a git repository and add this file to it. (Ask the TA if you don't remember how to do this from the last lab.)

3. Now go into the code and remove the parentheses to the right of 32 and rerun this code. Note the error returned. Fix the error and rerun the code `ruby -cw temp_conversions.rb`. The `-c` flag means 'check syntax only' and the `-w` flag activates higher warnings. When you run this time, you should see a new message – write it in the space below:

4. Add the test `puts convert("zero")` to the tests and rerun. Why did you get an error? To correct this, we will limit all temperatures to integers by adding a line before the calculation: `return "Temperature must be an integer" unless temp.class == Fixnum`. Rerun these tests after adding this line. If the tests pass, add the revision to the git repository. Also notice that if I want to return before the end of method, I need to use the `return` keyword, but if the method runs its course, the last value is automatically returned by the method. Finally, notice that I can place the condition at the end of the line in Ruby, not just at the beginning (as we are used to in other languages).

5. Add the test `puts convert(-500)` to the tests and rerun. Of course, remembering your basic physics leaves you distressed at this point because you know this answer is in error – Absolute Zero is at –474 ^{o}F or -270 ^{o}C, making this result impossible. To make sure our program doesn't give silly answers, we will add another line after the last correction (and before the calculation): `return "Temperature below Absolute Zero" if temp < -474` and rerun these tests. Assuming they pass, save the revision to the git repository.

6. Of course, we have only half the temperature conversion problem – converting Fahrenheit to Celsius – and have no capability to convert Celsius to Fahrenheit. Create a new branch in git called "exp" (Again, ask a TA if you don't remember this from previous labs). Now in your code, add another argument called 'measure' and using an `if ... else ... end` construct, correct the code so that either a Fahrenheit or Celsius temperature is converted. Set up the 'measure' argument so its default value is "F". Add the test below and rerun the code; if they pass, save to the repository.

```
puts convert(0, "C")
puts convert(10, "C")
puts convert(100, "C")
puts convert(-280, "C")
```

7. Looking at the results, we see that the code is still problematic: we get a result for –280 °C even though we know that value is below Absolute Zero. There are a number of ways to correct this, but for learning purposes here, we are going to create a new method called `below_absolute_zero?` which has two arguments: `temp` and `measure`. This method will simply return a boolean of true if the temperature for the measurement system is below the critical value; this is why this method will end in a question mark. Create the basic structure for this method now.

   Inside the new method, add the following two lines of code:

```
return false unless (temp < -454 and measure == "F") or
    (temp < -270 and measure == "C")[1]
true
```

   Now go back to the `convert` method and change the if statement for the Absolute Zero condition so that it references this new method rather than the simple statement of `temp < -474`. Rerun the code and make sure that everything is working properly. If

[1] Ruby allows us to use the words 'and' and 'or' but we can also use && for 'and' and || for 'or'. I encourage you to replace those words with these symbols later and convince yourself this is true.

so, save this code to the git repository and then merge the exp branch back onto the master branch. (Again, ask a TA is you don't remember how to do this.) Have a TA check off that your code is running correctly and that your git repository is in order.

8. We are going to switch gears now and start to interact with Ruby with irb [interactive ruby]. The irb is a useful command line tool that can allow you to try out simple ruby constructs and methods. To get to interactive ruby, just type `irb` on the command line.

9. Once in irb, type `a = Time.now` to get an object `a` with a Ruby timestamp. Type in `b = Date.today` and compare it with the results of the first command. What is the difference between these two objects? Write your answer below:

10. Now type `a.month` and note what is output is. Now type `a.strftime("%B")` and note the output. How would you find the year of `a`? (*Guess …*) Write your answer below and then test it.

11. Now type the command `a.month == b.month`. What was returned? Type the command `a.day == b.day - 1` and see that result. What is Ruby doing here? Write your answer below:

12. Use the `strftime` method to format the object a in the format of 99/99/9999. To see the options with `strftime` we can take advantage of a gem we installed in lab 1: cheat. In another

terminal window/tab, type `cheat strftime` and see the output returned – this should guide you in reformatting the timestamp.[2]

13. We are going to add three variables at once through multiple assignment (an option in Ruby). To do this, type into irb: `m1, d1, y1 = 04, 01, 2012`. Now we will use these values to create a new timestamp in Ruby with the command: `c = Time.mktime(y1, m1, d1)`
    Now we have the timestamp for this year's April Fool's Day. Let's change it a little bit by typing `d1 = 31` and rerunning the command: `c = Time.mktime(y1, m1, d1)`. What did Ruby do with the date 04/31/2012? Write your answer below:

On Your Own…

If you have time remaining now and still need to complete the SQL autograder (http://rook.hss.cmu.edu/~67272/SQL_Lab.php), you should do so now when TAs are here to help. Otherwise, this week the "on your own" assignment is to go to RubyMonk at http://rubymonk.com/ and complete

• exercise 0 -- Introduction to Arrays: [http://rubymonk.com/chapters/1-collection/lessons/2-arrays-introduction] and

• exercise 1 -- Introduction to Objects: [http://rubymonk.com/chapters/6-ruby-objects/lessons/35-introduction-to-objects].

This will give you more familiarity with Ruby so you will be prepared when you see Ruby code in a Rails project.

pItlh

[2] There are a lot of cheats available to you with this gem. To see the complete list, type `cheat sheets`. For a good laugh, try `cheat taxes` or `cheat girlfriend`. You can even `cheat ruby` to see some of the flags like –c we used earlier, or `cheat irb` or `cheat rails` to see what general options are available plus get a reference to other cheats. A word to the wise: learning what options are available as cheats can be helpful in later labs and on assignments.