CS 61A Structure and Interpretation of Computer Programs
Fall 2016 Midterm 2 Solutions

1. **(2 points) Route Cipher**

   A full implementation for `RouteCipher` can be found in **this Python script.** Don't worry too much about understanding all the details.

2. **(8 points) True or False**

   Is it `True`? Or is it `False`? You decide!

   (a) If you want to call a bound method, then you must explicitly pass in an argument as the `self` parameter.

   False. By definition, a bound method already has an instance passed in as its first parameter.

   (b) You can define a normal function (i.e. the kind we've been using all year) within a class, and access it without the use of dot notation.

   False. A function defined within a class must be accessed using dot notation – even from other functions within the class!

   (c) A name defined within a bound method will stick around for as long as the associated instance exists.

   False. This isn't always true. For example, a local variable (i.e. a non-"dot expression" variable defined inside a function) will only stick around for the lifespan of its frame.

   (d) All user-defined classes are technically subclasses.

   True. Everything inherits from `object`, either directly or through a chain of base classes.

   (e) If a function defined in a class takes `self` as its first argument, then it must always be called using dot notation with an *instance* on the left side of the dot.

   False. You can call such a function with a *class* on the left side of the dot, although you will then be forced to pass in a specific instance as `self`.

   (f) `self` is a special name in Python. If you were to use, say, `myself` as a method's first parameter name, then things would break.

   False. You can technically call this parameter something else. By convention, however, we use `self`.

   (g) If you change something in a subclass, then that change will propagate to all instances of the base class.

   False. It's kind of the other way around; changes to a *base class* can affect all *subclasses*.

   (h) In general, it's fine to replace an instance on the left side of a dot expression with `self`.

   False. You can only use `self` if it's actually been defined (inside a constructor or method, probably).