Julia Daniel
CS 106A

Section #1
April 11, 2018

Section #1: Lesson Plan

Written by Julia Daniel

In the first week of section, we're hoping you can get to know your section and they get to know each other, and that they understand why section is such a cool and important part of 106A. In addition, this will be the only section that covers Karel, so we hope you can get through at least the United Nations Karel problem.

Getting to Know Your Section

Feel free to spend a good chunk of time learning names, establishing how section works, and answering logistical questions about the class. Fun icebreakers are a plus. Make it clear that you are here to support them in their learning over the course of 106A. Remind them that you are their primary point-person for questions about the class (although post on Slack or get in touch with Julia if you ever aren't sure how to answer a particular question from a student!). Introduce IGs as a one-on-one time to talk about their assignments and about how the class is going generally. You may have IGs before you next see them, since Karel grading and IGs are due next Friday, so this is an important point to make. All this being said, if students go down rabbit holes of detailed logistical questions, feel free to have them come talk to you after section so that you leave at least ~20 minutes for Karel.

Lecture Recap

So far in 106A, there have been 5 lectures, and this section covers the first 3 (in **bold**). Please check out the slides in full if you haven't been in lecture, but here's a quick recap:

1. **Intro to Karel**
   Course logistics, 4 basic Karel commands, anatomy of a program: import statements + class definition + run method + helper methods (Chris referred to this as "adding vocabulary words" to Karel's vocabulary)
2. **Control flow in Karel**
   `for`, **`while`**, **`if-else`**, and running programs in Karel (direction testing, etc.)
3. **Decomposition**
   Break down a problem into milestones, then use top-down decomposition to get to a particular milestone, test and fix, then move on to next milestone
4. Variables
   Variables as named boxes that you can put values into and read values from
5. Control flow in Java
   Booleans, operator precedence, using for loop variables

Karel Problem(s)

Students will be wrapping up their time with Karel after this week (except for the midterm!), but we've given two problems that should help solidify some of the important concepts they need for approaching future assignments, specifically **milestoning**, **problem decomposition**, and **testing**. One major point to make is that ***finding bugs is a sign of achievement, not failure*** – no one writes perfect code on the first try all the time, but *finding* your bugs quickly, and writing code that leads to *simpler, easy-to-fix* bugs, are skills to learn. Feel free to share your own past frustrations with buggy code with students, especially later on in the quarter with students who feel discouraged or "bad at coding".

United Nations Karel

There are two main milestones to tackle, and the order doesn't really matter:

- Building a single house
  - Assuming we know we are at a location where we want to build a house (abstracting away how we got there), how do we do it?
  - Talk about decomposing repeated tasks – in this case, making a column of three beepers – and adding "vocabulary items" (methods) to Karel.
- Crossing the entire world and putting a house wherever you find rubble
  - Talk about abstracting away ***how*** the house is built to just focus on telling Karel ***when*** to build a house; assume you have a working **`buildHouse()`** method.
  - Test that this can be done correctly, without worrying about whether the house itself is built correctly. This can just be a couple sentences – *"let's run through this. If we have rubble here and here, will this [pseudo]code put houses where we want?"* – to build in habits in your sectionees of testing whenever they think they've finished coding a milestone.

Karel Defends Democracy

Here, too, think about how to incorporate lessons of milestoning, decomposition, and testing. If you don't make it through this problem, no worries. Make sure to tell students that we give them more problems than they'll get to in section on purpose – these are good practice problems for them to tackle on their own!