Eric Roberts and Jerry Cain
CS 106J
Handout #6
April 7, 2017

Assignment #1

Due: 5:00 P.M., Monday, April 17

Part I—Send us mail

Send an introductory e-mail message to your section leader (as soon as you know who it is), your instructors Eric Roberts (`eroberts@cs.stanford.edu`) and Jerry Cain (`jerry@cs.stanford.edu`) and your TA Jason Chen (`cheson@stanford.edu`). Please make sure that the subject line of your message says "CS106J Email: <*your name*>", where "<*your name*>" is actually filled in with your name. Here's the information to include in your e-mail:

1. Your name
2. Your year (frosh, sophomore, junior, senior, graduate, other)
3. Your major or area of interest ("unknown" is a perfectly acceptable answer)
4. Why you decided to take CS 106J
5. What you are most looking forward to about the class
6. What you are least looking forward to about the class
7. Any suggestions that you think might help you learn and master the course material

And if you feel like giving us a little more to remember you by, you could also include the following:

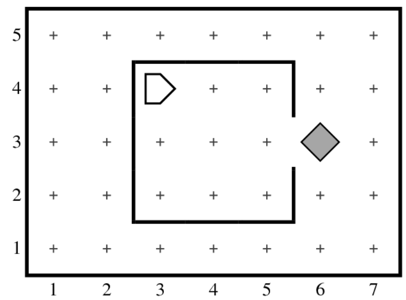
8. What do you do for fun?
9. Tell us a quick anecdote about something that makes you unique—a talent, an unusual experience, or anything of that sort.

Part II—Solve some Karel problems

The real problem-solving part of this assignment consists of four Karel programs. The starter project on the CS 106J web site contains the starter code for all four of these programs. When you want to work on one of these programs, you need to download the Assignment #1 starter folder as described in Handout #5 (Using JSKarel). From there, you need to edit the program files for each problem so that your program does what it's supposed to do, which will involve a cycle of coding, testing, and debugging until everything works.

Problem 1

Your first task is to solve a simple story-problem in Karel's world. Suppose that Karel has settled into its house, which is the square area in the center of the following diagram:



Karel starts off in the northwest corner of its house as shown in the diagram. The problem is to program Karel to collect the newspaper—represented (as all objects in Karel's world are) by a beeper—from outside the doorway and then to return to its initial position.

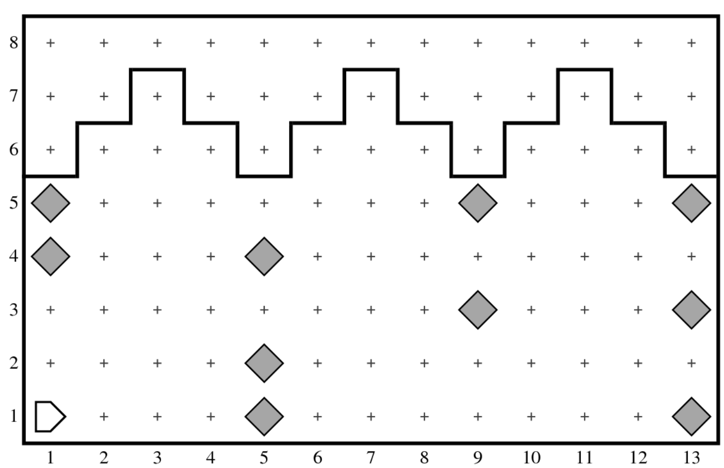
This exercise is extremely simple and exists just to get you started. You can assume that every part of the world looks just as it does in the diagram. The house is exactly this size, the door is always in the position shown, and the beeper is just outside the door. Thus, all you have to do is write the sequence of commands necessary to have Karel

1. Move to the newspaper.
2. Pick it up.
3. Return to its original starting point.

Even though the program requires just a few lines, it is still worth getting at least a little practice in decomposition. In your solution, include a helper function for each of the steps shown in the outline.

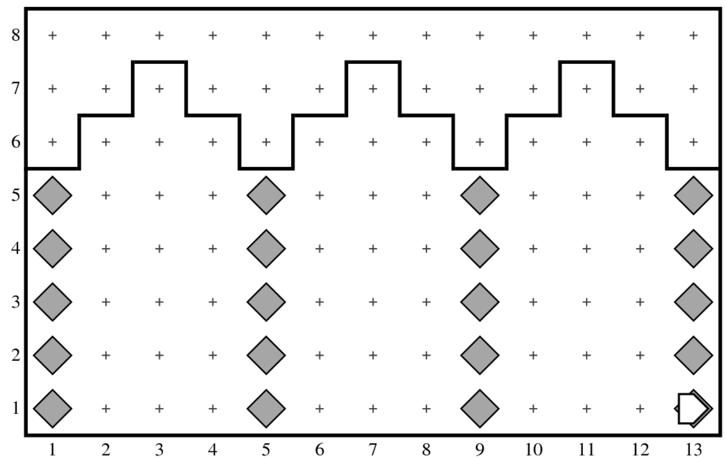
Problem 2

Karel has been hired to repair the damage done to the Quad in the 1989 earthquake. In particular, Karel is to repair a set of arches where some of the stones (represented by beepers, of course) are missing from the columns supporting the arches, as follows:



Your program should work on the world shown above, but it should be general enough to handle any world that meets certain basic conditions as outlined at the end of this problem. There are several example worlds in the starter folder, and your program should work correctly with all of them.

When Karel is done, the missing stones in the columns should be replaced by beepers, so that the final picture resulting from the world shown above would look like this:

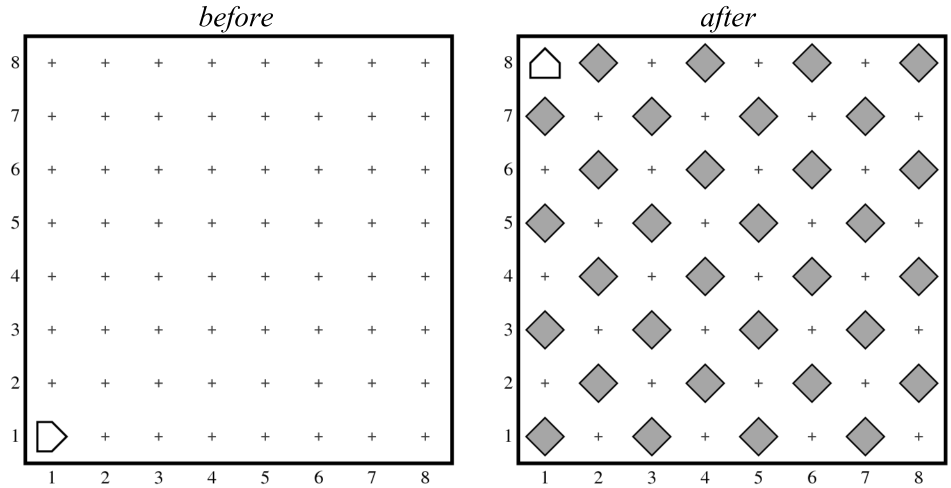


Karel may count on the following facts about the world:

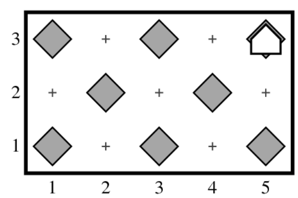
- Karel starts at 1st Avenue and 1st Street, facing east, with an infinite number of beepers.
- The columns are exactly four units apart, on 1st, 5th, 9th Avenue, and so forth.
- The end of the columns is marked by a wall immediately after the final column. This wall section appears after 13th Avenue in the example, but your program should work for any number of columns.
- The top of the column is marked by a wall, but Karel cannot assume that columns are always five units high, or even that all columns are the same height.
- Some of the corners in the column may already contain beepers representing stones that are still in place. Your program should not put a second beeper on these corners.

Problem 3

In this exercise, your job is to get Karel to create a checkerboard pattern of beepers inside an empty rectangular world, as illustrated in the following before-and-after diagram:



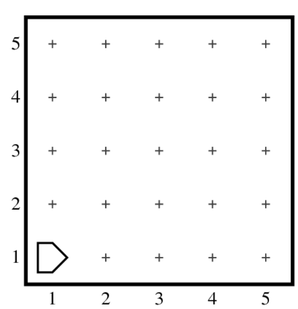
This problem has a nice decomposition structure along with some interesting algorithmic issues. As you think about how you will solve the problem, you should make sure that your solution works with checkerboards that are different in size from the standard 8x8 checkerboard shown in the example. Odd-sized checkerboards are tricky, and you should make sure that your program generates the following pattern in a 5x3 world:



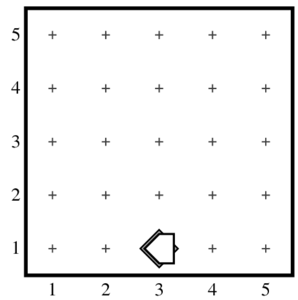
Another special case you need to consider is that of a world which is only one column wide or one row high. The starter folder contains several sample worlds that test these special cases, and you should make sure that your program works for each of them.

Problem 4

As an exercise in solving algorithmic problems, program Karel to place a single beeper at the center of 1st Street. For example, if Karel starts in the world



it should end with Karel standing on a beeper in the following position:



Note that the final configuration of the world should have only a single beeper at the midpoint of 1st Street. Along the way, Karel is allowed to place additional beepers wherever it wants to, but must pick them all up again before it finishes.

In solving this problem, you may count on the following facts about the world:

- Karel starts at 1st Avenue and 1st Street, facing east, with an infinite number of beepers in its bag.
- The initial state of the world includes no interior walls or beepers.
- The world need not be square, but you may assume that it is at least as tall as it is wide.

Your program, moreover, can assume the following simplifications:

- If the width of the world is odd, Karel must put the beeper in the center square. If the width is even, Karel may drop the beeper on either of the two center squares.
- It does not matter which direction Karel is facing at the end of the run.

There are many different algorithms you can use to solve this problem. The interesting part of this assignment is to come up with a strategy that works.