

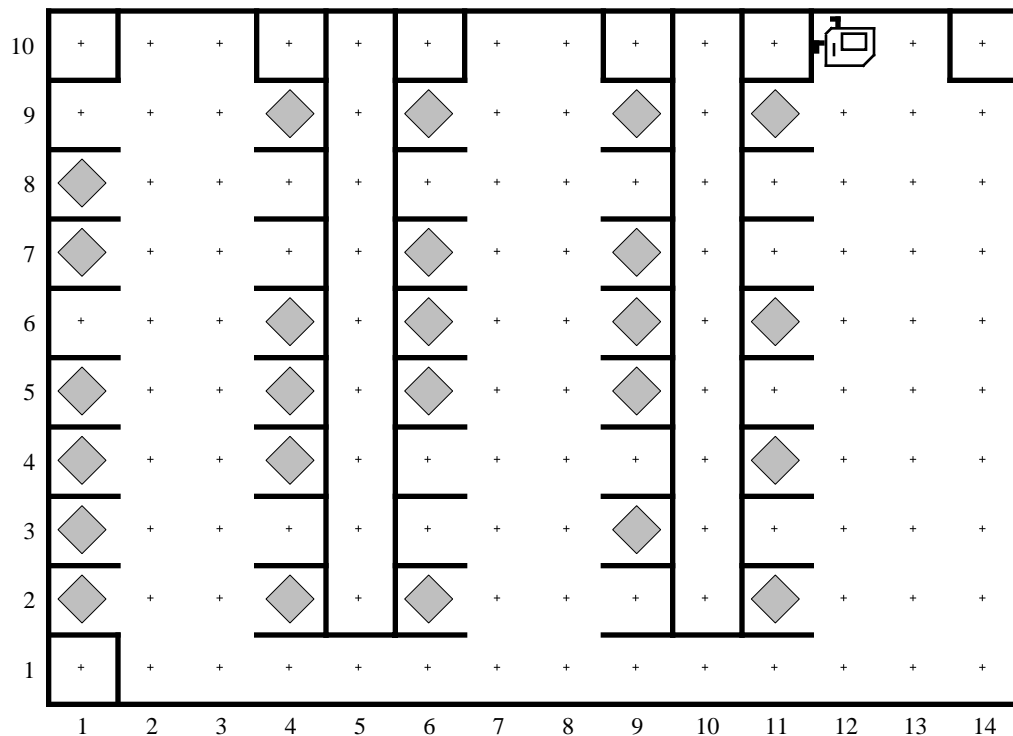
## Section Handout #1—Karel and Simple C

This week in section, your first priority is to meet your section leader and discover what sections in CS106A are all about. Your section leader will therefore spend the first part of this week’s session on introductions and telling you the things you need to know, such as where to get help and how to sign up for interactive grading. After the introductory material, the section will move on to cover some of the important material from class in a setting that is small enough for you to ask questions and thereby find out what you need to know. This week, your goals are to solve a substantial Karel problem involving stepwise refinement and to get started with some of the simple C constructs I will introduce today in class.

Most students in CS106A find that they don’t have too much trouble understanding the basic operations in Karel’s world or even how to use the Karel application. Karel’s world is, after all, quite constrained and it isn’t usually too hard to understand any one single part of it. Even so, it is a good idea for you to try solving this problem before you go to section so that you know what questions to ask. The fact that the individual operations in Karel’s world are simple does not imply that programming Karel to solve problems will be easy. The difficulty of a program depends on the complexity of the problem: writing programs to solve difficult problems is hard. Some of you will find Assignment #1 challenging and will need to use strategies like stepwise refinement to simplify each task.

### Problem 1. Karel at helper hours

As you will soon discover, one of the reasons that CS106A has been so effective is that the section leaders staff the helper cubicle in the Tresidder Lair. This year, let’s imagine that a group of enterprising section leaders have decided to automate the process by programming Karel the Robot to staff the help students in the class. In Karel’s world, the Lair cluster looks something like this:

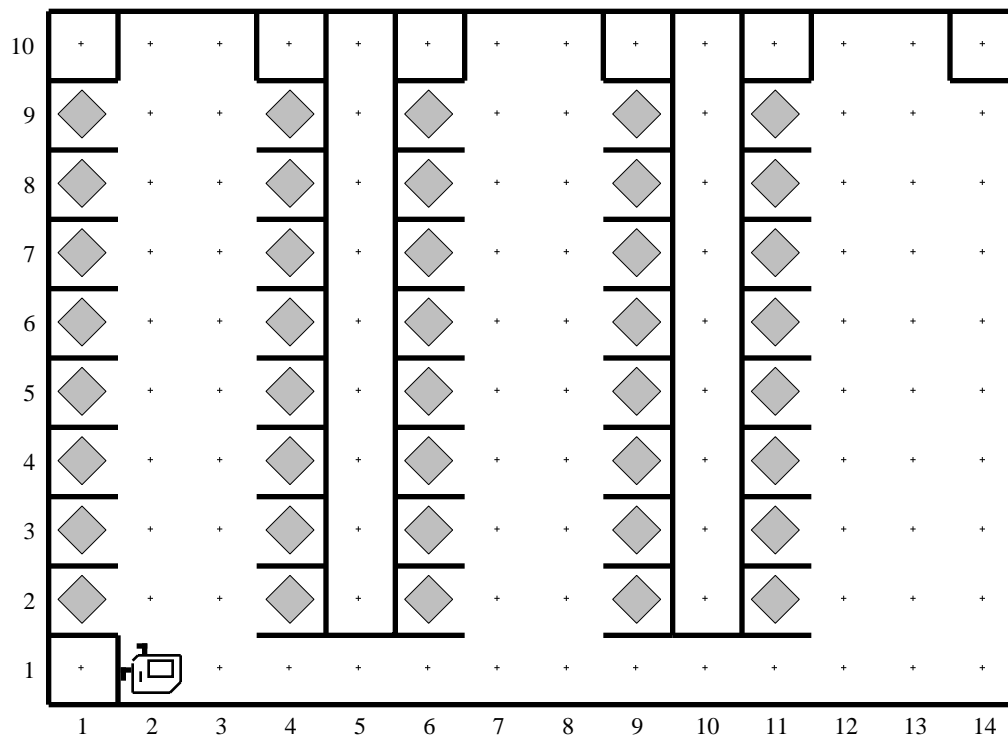


Karel starts out in the helper cubicle facing south in the position shown. The plan is to program Karel to walk through each row of computers and help any student who needs it. In this diagram, the students who need help are indicated by the *empty* computer stations. The computer stations that already contain beepers indicate students working on other courses for which Karel has no help to offer. Karel's task is then to visit all the computer stations and dispense pearls of wisdom—represented by beepers of course—to every empty one.

In solving this problem, Karel may count on the following facts:

1. Karel starts in the helper cubicle facing south, exactly one corner to the north of the first computer station.
2. Karel's bag starts out with an infinite number of beepers.
3. Karel does not know how many computer stations there are in each row, but does know that the stations start and end exactly one corner away from the boundary wall.
4. Karel does not know how many rows of computers there are in the room, but can count on the open corridors separating the individual stations being exactly two corners wide. Moreover, the stations in the middle of the room are arranged so that there is one corner of separation between the back of one cubicle and the back of the adjacent cubicle in the next row. Both of these properties are illustrated in the diagram of Karel's sample world.
5. The last row of computer stations is against the west wall of the room from which there is no corridor to any additional rows. Thus, Karel knows that the task is finished when it is impossible to move west at the bottom of a row.

The final configuration for Karel therefore looks like this:



Write the Karel program necessary to deliver the necessary help to every terminal station in the Lair that needs it.

## Problem 2. Converting Fahrenheit to Celsius

Because we start our discussion of C on Wednesday, it is useful, assuming there is time in section, to think about simple C programs and how their structure is similar to and distinct from that of Karel. The syntax of the program file has many similarities, such as the braces and the appearance of a procedure named `main`, which constitutes the program. Statements in C, however, work in the context of a different conceptual environment, where the individual actions are no longer things like `Move` or `TurnLeft`, but instead involve the manipulation of data values stored in variables.

To give yourself a chance to play with these C programs, try the following exercise from Chapter 2 (page 56):

7. Write a program that reads in a temperature in degrees Fahrenheit and returns the corresponding temperature in degrees Celsius. The conversion formula is

$$C = \frac{5}{9}(F - 32)$$

The following is a sample run of the program:

```
Program to convert Fahrenheit to Celsius.  
Fahrenheit temperature? 212↵  
Celsius equivalent: 100
```