Problem 1 (`CollectNewspaperKarel.py`)

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 Figure 1. During this time, you might correctly surmise that Karel is sheltering-in-place.

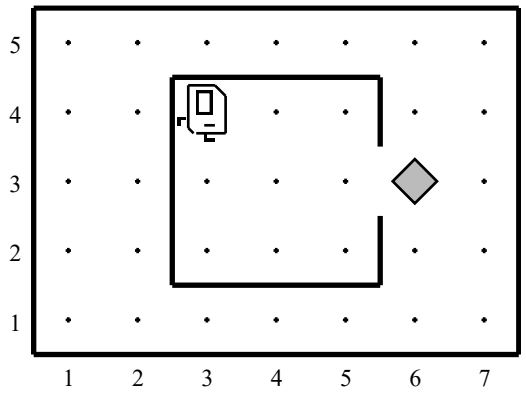

Figure 1: Karel's starting state for **`CollectNewspaperKarel`**

Karel starts off in the northwest corner of its house as shown in the diagram. The problem you need to solve is to get Karel to collect the newspaper. The newspaper, like all objects in Karel's world, is represented by a beeper. You must get Karel to pick up the newspaper located outside the doorway and then to return to its initial position.

This exercise is simple and is meant to help you get you started programming with Karel. 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, and
3. Return to its starting point.

Although the program does not have many lines of code, it is still worth getting some practice with decomposition. In your solution, include a function for *each* of the three steps shown in the outline above.

Your program should run successfully in the following world:

`CollectNewspaperKarel.w` (default world)

Note that all Karel worlds are located in the **`worlds`** folder in the Assignment 1 project folder.

Problem 2 (`TripleKarel.py`)

Your second task is to help Karel paint the exterior of some oddly-shaped buildings using beepers! For this problem, Karel starts facing west next to a "building" (represented by a rectangle, constructed from walls) whose sides span one or more street corners. Karel's goal is to paint all of the buildings present in the world by placing beepers along three of the sides of each of the buildings.

We recommend breaking down the problem into the following steps:

1. First, Karel should paint one side of the rectangle, placing beepers on all corners that are adjacent to the wall of the building. Note that there's a boundary detail here: **the last square where Karel ends should not have a beeper on it**.
2. Next, Karel should accomplish the task of painting a single rectangle. Think about how you can use the functionality of the previous subtask to help you accomplish this goal. You may need to write a small amount of code to reposition Karel in between painting individual walls of a building.
3. Finally, the overall `TripleKarel` problem is just painting all three buildings in the world. Again, you may need to write a small amount of code to reposition Karel in between painting individual buildings.

Figures demonstrating the before and after stages of each of the three steps are shown on the following page.

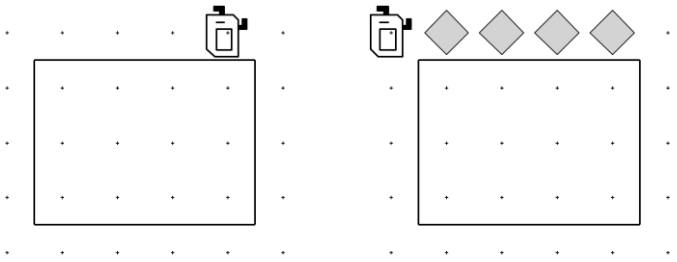

Figure 2: After you've completed the first step, Karel should be able to paint one side of one building. Running the program would result in the above start (left) and end (right) states.

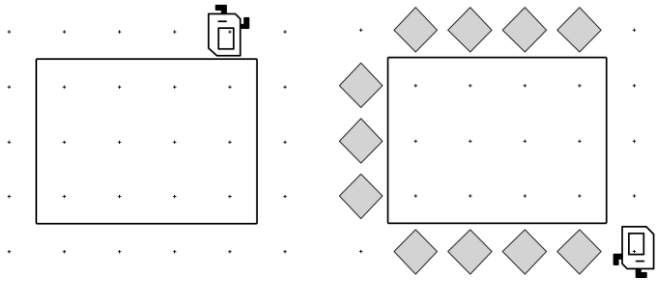

Figure 3: After you've completed the second step, Karel should be able to paint one building. Running the program would result in the above start (left) and end (right) states.

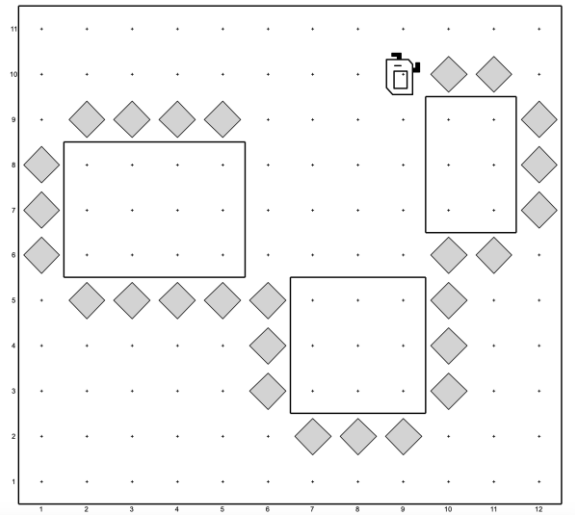

Figure 4: After you have painted the whole world, the end result should look like this.

You can assume that:

- Karel will always start facing west at the upper right corner of the leftmost building (at the position where the first beeper should be placed).
- Karel will have infinite beepers in his beeper bag, so he can paint any size of buildings.
- Although buildings may be of varying sizes, there will always be exactly three of them, and their relative position to one another will always be the same (as displayed in Figure 6). If you are still confused about what assumptions you can make about the world, see the additional **`Triple`** world files we have included.

You should make sure your program runs successfully in all of the following worlds (which are just a few different examples to test out the generality of your solution):
`TripleKarel.w` (default world), **`Triple1.w`**, **`Triple2.w`**, **`Triple3.w`**

Note that all Karel worlds are located in the **`worlds`** folder in the Assignment 1 project folder.

Problem 3 (`StoneMasonKarel.py`)

Your third task is to repair the damage done to the Main Quad in the 1989 Loma Prieta earthquake. In particular, Karel should repair a set of arches where some of the stones (represented by beepers, of course) are missing from the columns supporting the arches, as illustrated in Figure 5 (below).

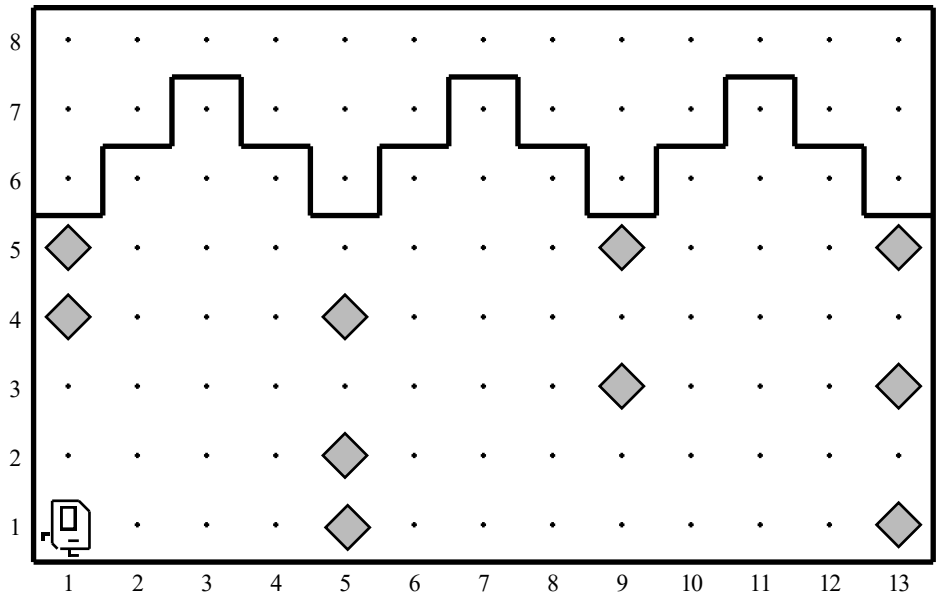

Figure 5: An initial example world with broken arches that **`StoneMasonKarel`** must repair

Your program should work on the world shown above, but it should be general enough to handle any world that meets the basic conditions outlined at the end of this problem. There are several example worlds in the starter folder, and your program should work correctly in 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 initial world shown in Figure 5 would look like the illustration in Figure 6.

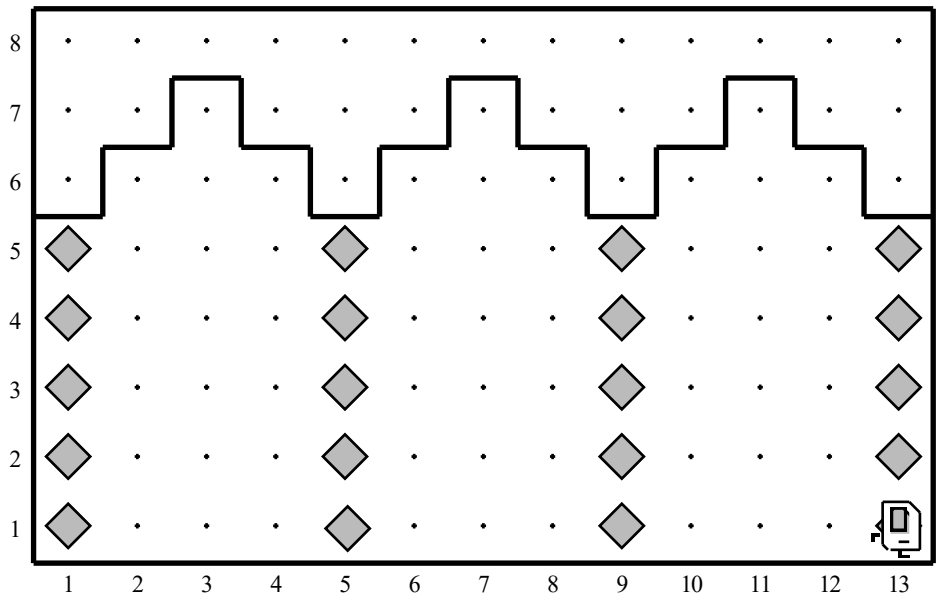

Figure 6: Karel should repair the Main Quad to a structurally sound state after completion.

Karel's final location and the final direction Karel is facing at the end of the run do not matter.

Karel may count on the following facts about the world:

- Karel starts at the corner where 1st Avenue and 1st Street meet, facing east, with an infinite number of beepers in Karel's beeper bag. The first column should be built on 1st Avenue.
- The columns are always exactly four Avenues apart, so they would be built on 1st Avenue, 5th Avenue, 9th Avenue, and so on.
- The final column will always have a wall immediately after it. Although this wall appears after 13th Avenue in the example figure, your program should work for any number of beeper columns.
- The top of a beeper column will always be marked by a wall. However, Karel cannot assume that columns are always five units high, or even that all columns within a given world are the same height.
- In an initial world, some columns may already contain beepers representing stones that are still in place. Your program should not put a second beeper on corners that already have beepers. Avenues that will not have columns will never contain existing beepers.

You should make sure your program runs successfully in all of the following worlds (which are just a few different examples to test out the generality of your solution):
`StoneMasonKarel.w` (default world), **`SampleQuad1.w`**, **`SampleQuad2.w`**

Note that all Karel worlds are located in the **`worlds`** folder in the Assignment 1 project folder.

Problem 4 (`CheckerboardKarel.py`)

Your fourth and final task is to get Karel to create a checkerboard pattern of beepers inside an empty rectangular world, as illustrated in Figure 7. (Karel's final location and the final direction it is facing at the end of the run do not matter.)

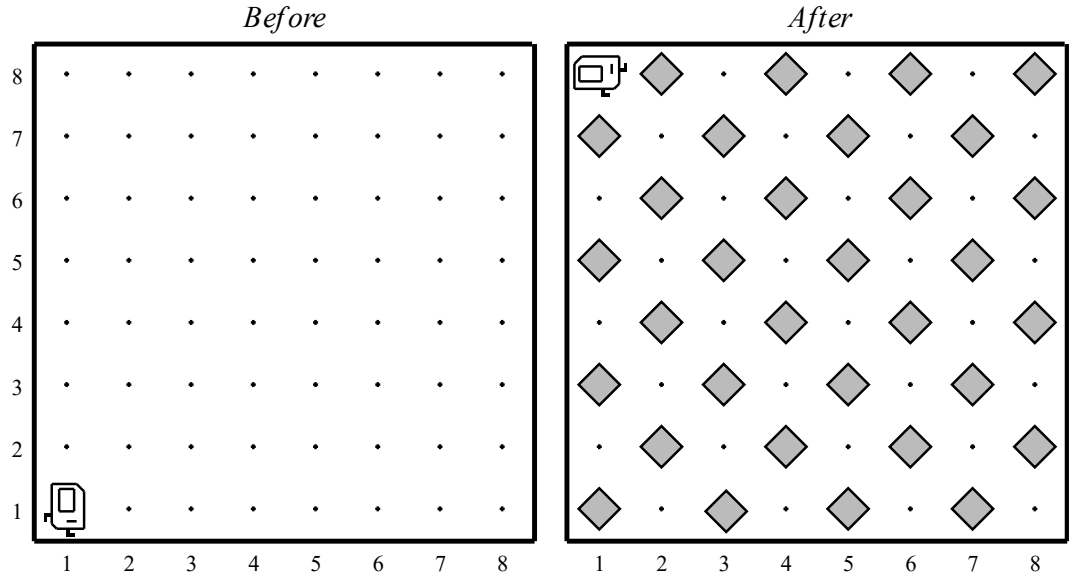

Figure 7: The beginning and end states for **`CheckerboardKarel`**.

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 above. Some examples of such cases are discussed below.

Odd-sized checkerboards are tricky, and you should make sure that your program generates the following pattern in a 5x3 world:

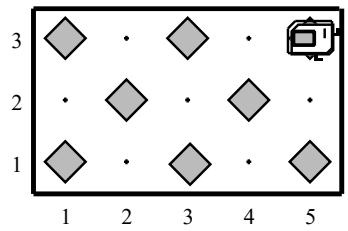

Figure 8: Karel should generate this checkerboard pattern for a 5x3 world.

Other special cases you should consider are worlds with only a single column or a single row. The starter code folder contains several sample worlds with these special cases, and you should make sure that your program works for each of them.

This problem is hard: Try simplifying your solution with decomposition. Can you checker a single row/column? Make the row/column work for different widths/heights? Once you've finished a single row/column, can you make Karel fill two? Three? All of them? Incrementally developing your program in stages helps break it down into simpler parts and is a wise strategy for attacking hard programming problems.

You should make sure your program runs successfully in all of the following worlds (which are just a few different examples to test out the generality of your solution):
`CheckerboardKarel.w` (default world), **`8x1.w`**, **`1x8.w`**, **`7x7.w`**, **`6x5.w`**, **`3x5.w`**, **`40x40.w`**, **`1x1.w`**

Note that all Karel worlds are located in the **`worlds`** folder in the Assignment 1 project folder.

Submission

Following the instructions in Handout #6 (Submitting Assignments), you should submit the following files (do not include any files not included in this list!):

- **`CollectNewspaperKarel.py`**
- **`TripleKarel.py`**
- **`StoneMasonKarel.py`**
- **`CheckerboardKarel.py`**

If you did the bonus portion(s) of the assignment, you should also submit:

- **`MidpointKarel.py`**
- **`ExtensionKarel.py`**