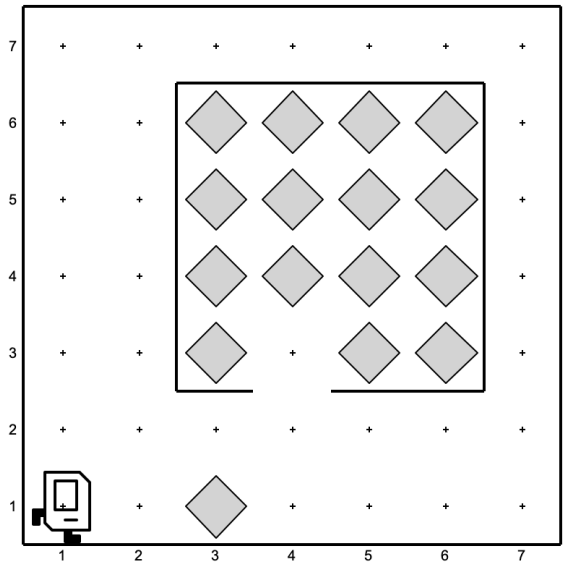
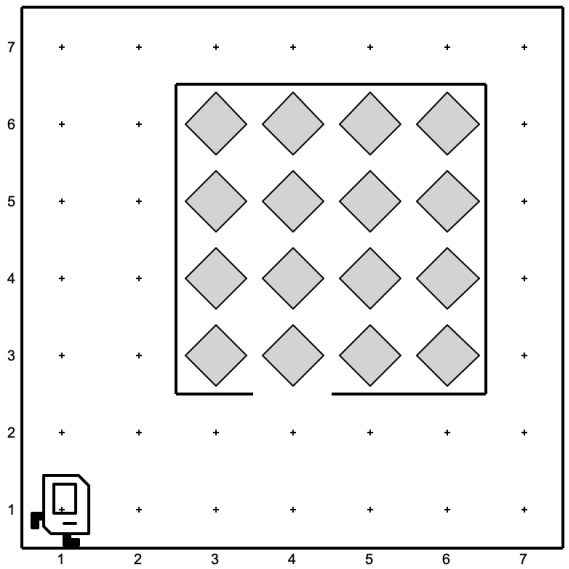
Assignment 1: Karel

Q1: Jigsaw Puzzle Karel

During quarantine, Karel has picked up a new hobby: doing puzzles! Karel is almost done with the square puzzle represented by the 4x4 grid of beepers shown below:



The beeper in the bottom most row represents the last piece of the puzzle! Write a program which will get Karel to pick up the last piece, put it in place, and move Karel back to the bottom left corner of the world facing East so she can admire the completed puzzle. Here's what your end result should look like:



To reiterate, you should write the sequence of commands so that Karel will:

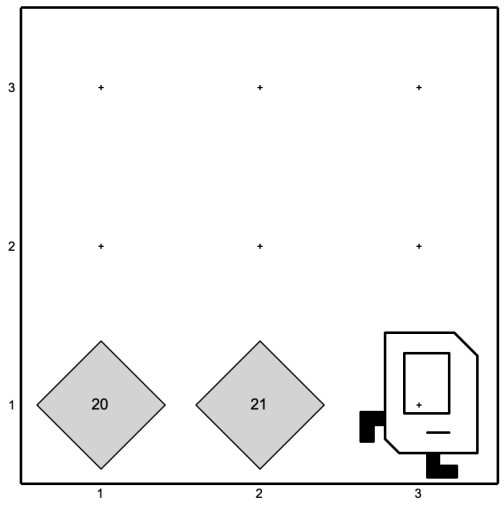
- Move to and pick up the last puzzle piece (the beeper in row 1, column 3)

- Put the puzzle piece in place (row 3, column 4)
- Return Karel to her initial position

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.

Q2: Year 2021

Congratulations on beginning your coding journey! Karel welcomes you to Code in Place 2021. Your next task is to help Karel celebrate the occasion by placing 20 beepers, moving Karel one step, placing 21 beepers, and moving Karel one more step. The world should ultimately look like this:



Happy coding!

Q3: Cleanup Karel, Milestone 1

Your next task is to execute a "safe pickup" -- Karel can pick up beepers, but not if none are present! Write a program which will check if a beeper is present at the position Karel is currently on and pick up a beeper if one is present (if there are no beepers present, Karel shouldn't do anything).

Two worlds are provided for you to test your code on -- on the world where Karel starts on a beeper, your code should get Karel to pick the beeper up. On the world where Karel starts on a blank spot, your code shouldn't do anything.

We've provided you two 1x1 worlds (one with a beeper, one without) on which to test your code. You can toggle from the beeper-present world to the no-beeper world by changing the very last line in the file from `run_karel_program('SafePickup1.w')` to `run_karel_program('SafePickup2.w')` (and vice versa).

For example, to use the `SafePickup1.w` world:

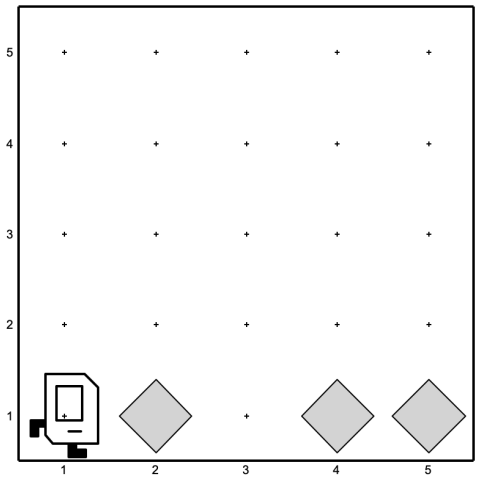
```
def main():
    # ... your code ...

if __name__ == '__main__':
    run_karel_program('SafePickup1.w')
```

Q4: Cleanup Karel, Milestone 2

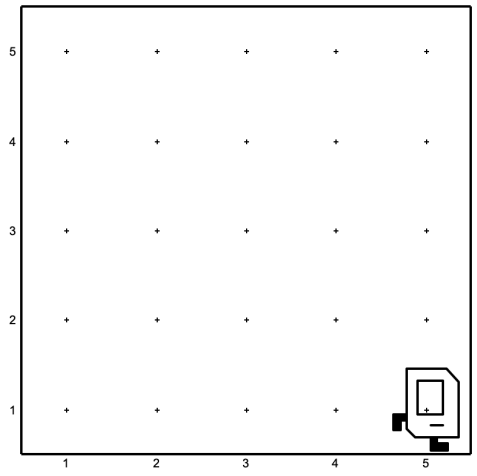
Karel has a bit of spring cleaning to do! Karel's world will have beepers in some positions in the bottom row; write a program to have Karel walk across the bottom row and, at each position, pick up a beeper only if one is present. Notice that you've already written the code to check if a beeper is present and only pick up a beeper if one is there from the previous milestone -- you should use your code from the previous milestone as a helper function to help with the decomposition of this problem!

Additionally, note that *Karel's starting position will never contain a beeper*, so there's no need to check it.

For example, if this is the initial starting world, with some beepers in the first row:



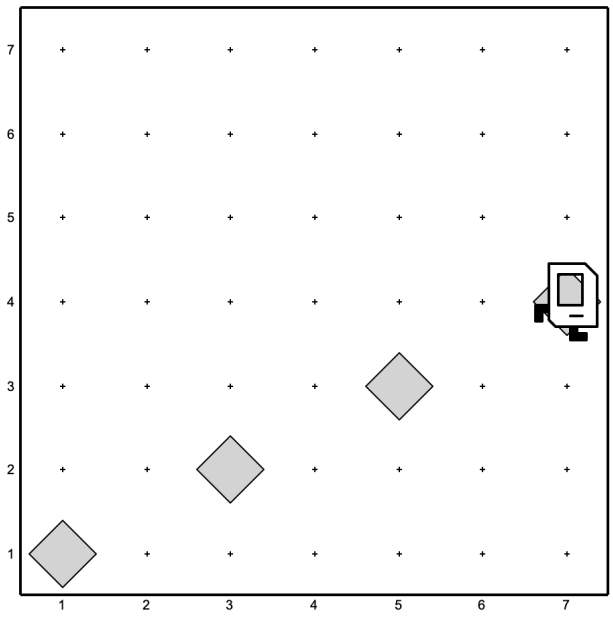
This should be the end result, with a clear bottom row:



We've provided you two worlds on which to test your code. You can toggle between them by changing the very last line in the file from `run_karel_program('Cleanup1.w')` to `run_karel_program('Cleanup2.w')` (and vice versa) -- you will likely need to press Run (it's fine if you do so without any code written) for the world change to take effect.

Q5: Ramp Climbing Karel

Write a program that has Karel draw a diagonal line across the world, with a slope of ½, like so:



The key to drawing a diagonal line with slope ½ is to move two steps forward and one step up between each beeper. In this problem you can and should assume that the world is an **odd number** of columns across. Solving the problem for even columns as well is much harder and would count as an "extension".

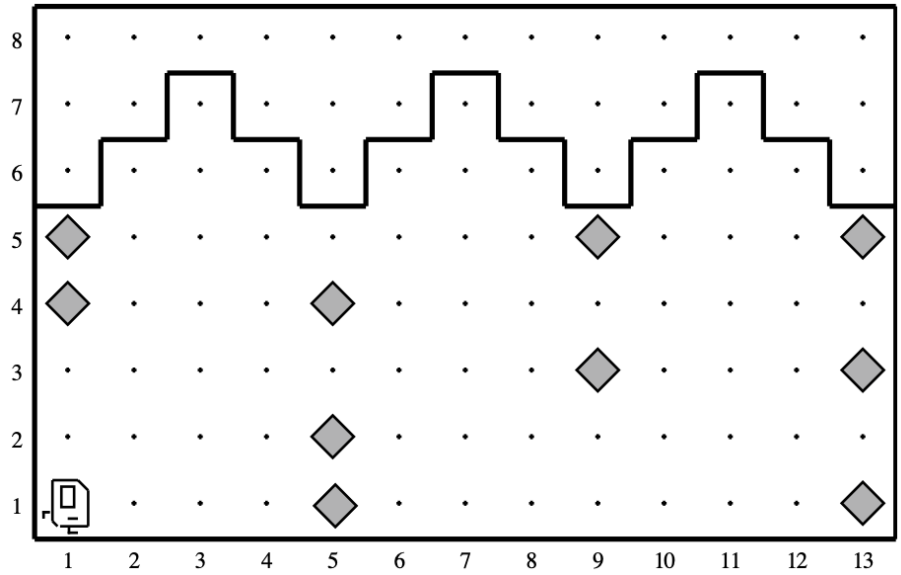
You should assume

- Karel **always begins** at the bottom left corner of and empty world facing East.
- You may assume that the world is an **odd number** of columns across
- Karel's bag has infinite beepers.
- It does not matter which direction Karel ends up facing.
- The world is always square (the world's height is the same as its width)

We've provided you three worlds on which to test your code. You can toggle between them by changing the very last line in the file from `run_karel_program('RampKarel1.w')` to `run_karel_program('RampKarel2.w')` or `run_karel_program('RampKarel3.w')` -- you will likely need to press Run (it's fine if you do so without any code written) for the world change to take effect. RampKarel1 is a 7x7 world, RampKarel2 is a 3x3 world, and RampKarel3 is a 25x25 world.

Q6: Stone Mason Karel

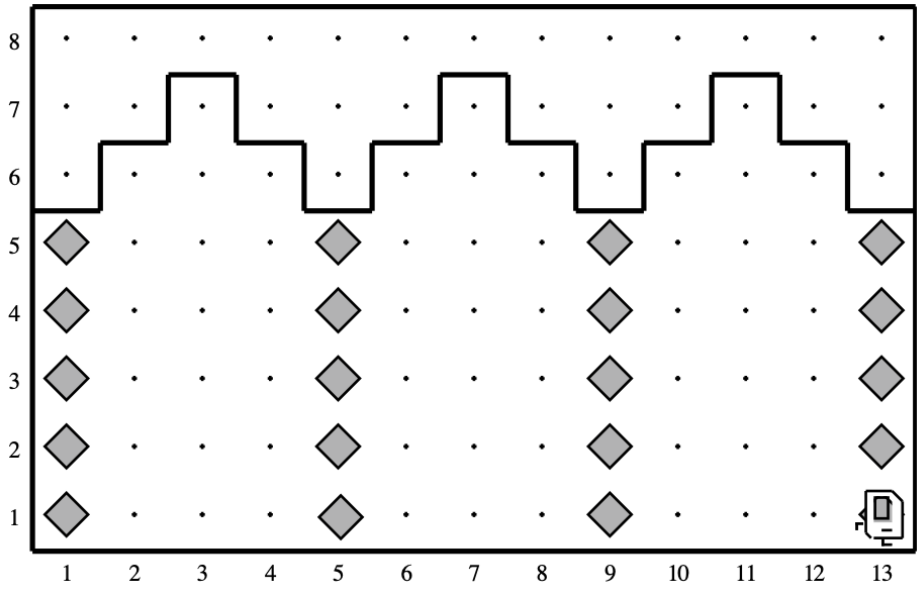
Your next task is to repair the damage done to the Stanford Main Quad in the 1989 Loma Prieta earthquake. In particular, Karel should repair a set of arches where some of the stones (represented by beepers, of course) are missing from the columns supporting the arches, as illustrated in the figure below.



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 three example worlds here, and your program should work correctly in all of them. You can toggle between them by changing the very last line in the file from `run_karel_program('SampleQuad1.w')` to `run_karel_program('SampleQuad2.w')` or `run_karel_program('SampleQuad3.w')` -- you will likely need to press Run (it's fine if you do so without any code written) for the world change to take effect.

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 below.



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.

Note: if you're missing the additional "1x1.w" and "21x21.w" world files, you can download these files below, and then add/upload them to your challenge workspace:

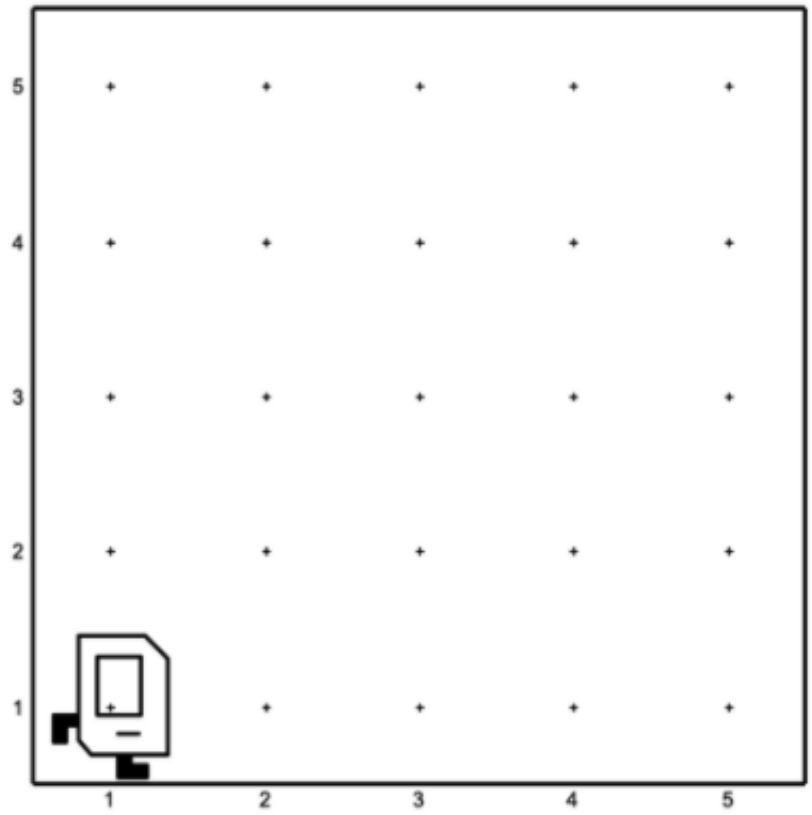
1x1.w

21x21.w

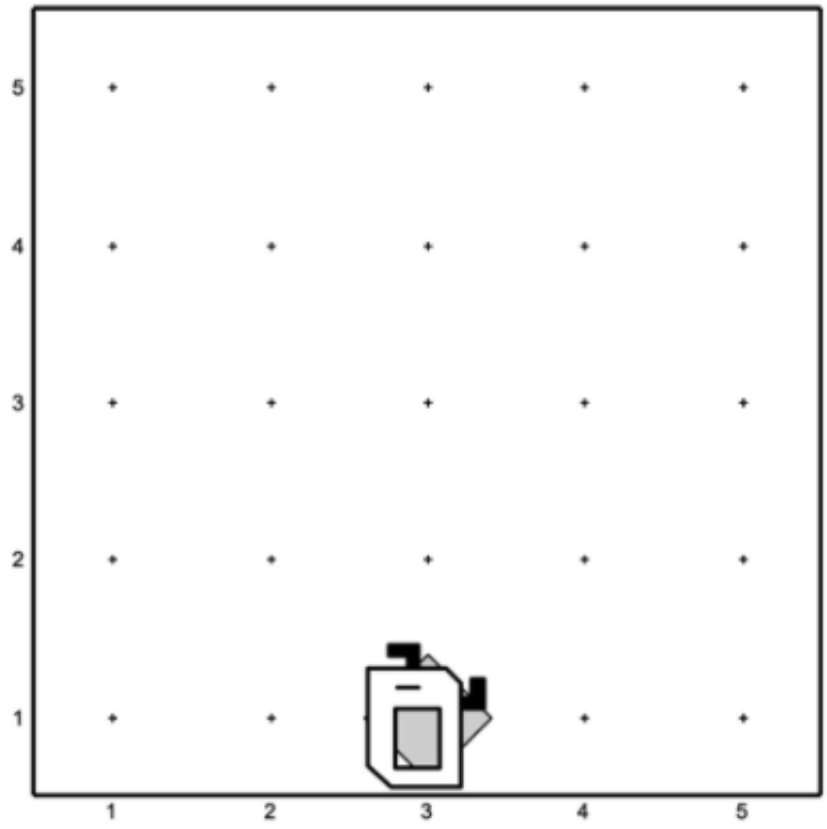
Alternatively, **save a copy of your code first**, and you can then use the ... > Reset to Scaffold in the top right to update automatically. All files will be reset back to their original state, so be careful.

Q7: Midpoint (optional!)

As an exercise in solving algorithmic problems, program Karel to place a single beeper at the **middle** of 1st Street (aka Row). For example, say Karel starts in the 5x5 world pictured in the figure:



Karel 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. Similarly, if Karel paints/colors any of the corners in the world, they must all be uncolored before Karel finishes.

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

- Karel starts at the bottom left corner, 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 so feel free to be creative!

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): `Midpoint.w` (default world), `Midpoint1.w`, `Midpoint2.w`, `Midpoint8.w` .

You can toggle between worlds by changing `Midpoint.w` in the last line of the file (which is currently `run_karel_program('Midpoint.w')` to the filename of your choice (make sure to include the quotation marks around the filename) and running your program.

Extension (optional!)

If you finish early, you may optionally write a Karel project of your own choice. Modify this file to use Karel to complete any task of your choosing! Extensions are a great chance for practice and to be creative. Make sure to write comments to explain what your program is doing and update the world to be appropriate for your program. (Notice that you can toggle the rows and columns of Karel's world, and if you right click Karel's world, you'll see a dropdown of other things you can do by clicking!)

Conceptual Q&A

Clarification Q&A