Karel HW 1

Problem 1 (CheckerboardKarel.py)

Your fifth and final task is to get Karel to create a checkerboard pattern of beepers inside an empty rectangular world, as illustrated in **Figure 1** . (Karel's final location and the final direction it is facing at the end of the run do not matter.)

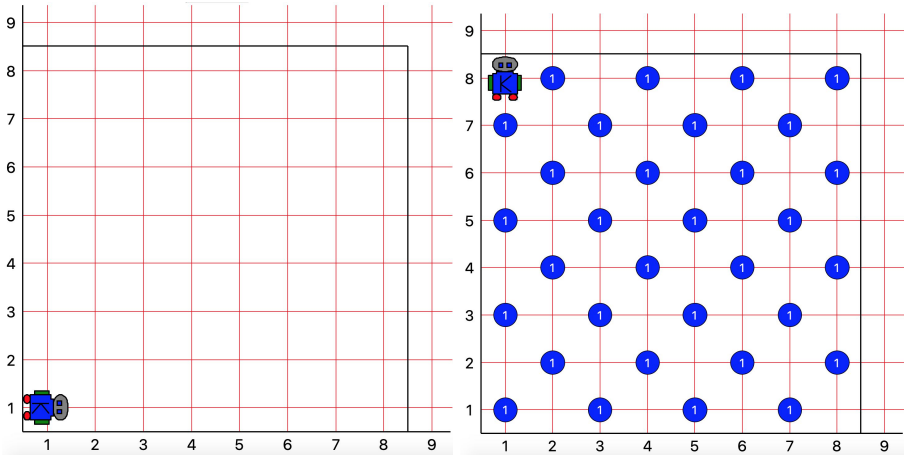

Figure 1 : The beginning and end states for **CheckerboardKarel** .

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:

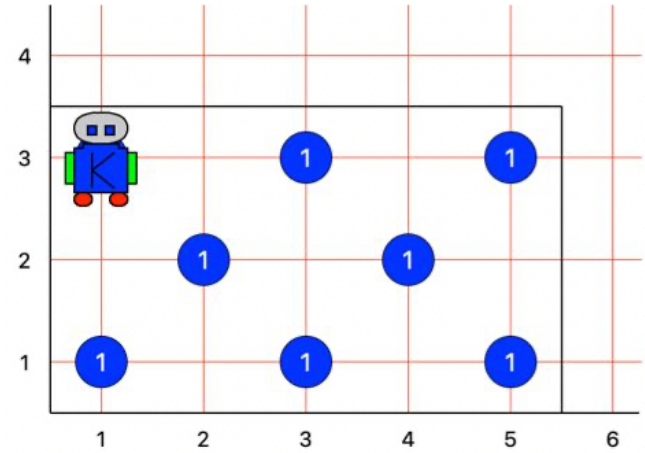

Figure 2 : Karel should generate this checkerboard pattern for a 5x3 world.

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.

Your program should run successfully in all of the following worlds (all worlds are located in the **worlds/** folder): **Checkerboard8x8.kwld** (default), **Checkerboard8x1.kwld** , **Checkerboard1x8.kwld** , **Checkerboard7x7.kwld** , **Checkerboard6x5.kwld** , **Checkerboard3x5.kwld** , **Checkerboard40x40.kwld** , **Checkerboard1x1.kwld**

Problem 2 (MidpointKarel.py - Bonus Project)

This problem is a bonus problem that is not required. Students who successfully complete this problem will be awarded a small amount of extra credit on the assignment.
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 a 5x5 world, it should end standing on a beeper as pictured in **Figure 3** .

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

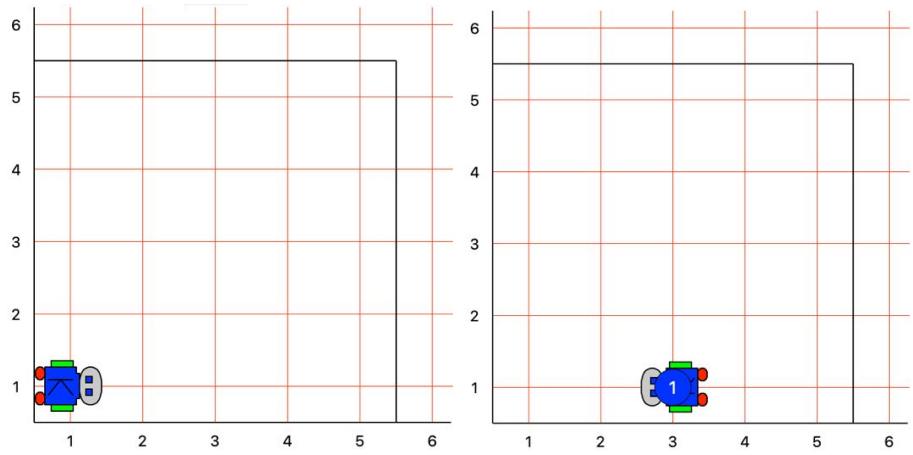

Figure 3: The beginning and end states for **MidpointKarel**

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

- Karel starts facing east at 1st Avenue and 1st Street.
- 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!

Your program should run successfully in all of the following worlds (all worlds are located in the **worlds/** folder): **Midpoint5.kwld** (default) **, Midpoint1.kwld , Midpoint2.kwld , Midpoint8.kwld**

Submission

The file you're expected to update are the file(s) with the *.py extension.

Submit a folder that is **only** containing this Python source file (*.py) at course location in NetStorage.

Please use the following naming convention for the submitted folder:

YourPSsection_CourseCode_Surname_Name_HWNumber_Semester

Example folder name:

- PSA_COMP125_Tuzmen_Ayca_HW1_S20

Additional notes:

- Using the naming convention properly is important; failing to do so may be penalized.
- Do not use Turkish characters when naming files or folders.
- Submissions with unidentifiable names will be disregarded completely.
  (ex. "homework0", "project" etc.).
- Please write your name into the source file where it is asked for:

Academic Honesty

Koç University's Statement on Academic Honesty and CS Honor Code holds for all the homework given in this course. Failing to comply with the statement will be penalized accordingly. If you are unsure whether your action violates the code of conduct, please consult with your instructor.