Karel The Robot Assignment

Atypon Java and DevOps Cohort / June 2024

Mahmoud J. Qudah

Introduction:

The Karel Assignment is an educational project that is quite challenging. There are many possible solutions, and you need to find a good one by practicing all the cases, including the edge cases, and trying to identify the best scenario for Karel to complete his task.

Since Karel is lost in his world and has no information about it, such as the width and height of the map, the task involves dividing this map into the largest number of chambers starting from four, three, two and one chamber (no dividing), by placing beepers where needed, the goal is to achieve this with the fewest number of moves and beepers used.

Explanation:

I believe I have come up with good solutions to help Karel achieve his goal, with more readable and reusable code to keep it as clean as possible, making it easier for other programmers to understand the process. I will explain how I handled all the cases and challenges.

First, I started by discovering the map to find its dimensions. Once I had them, I divided the problem into subproblems. Since I have four possible options, I created four different methods, each designed to handle a specific type of division.

Discovering the map:

I created two variables, width and height, and I always check if the path in front of Karel is clear. If it is, he moves forward, and I increase the width by one. When Karel faces a wall, I make him turn left and move upward to determine the height, this was the best way to start knowing the world with no need to walk around all the map, Once Karel calculates the dimensions, he returns home to decide how to start dividing.

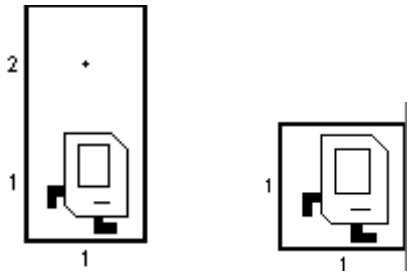
Let's start by explaining these four main methods that contain the core solutions:

1) One Room Dividing
2) Two Room Dividing
3) Three Room Dividing
4) Four Room Dividing

- ***One Room Dividing:***

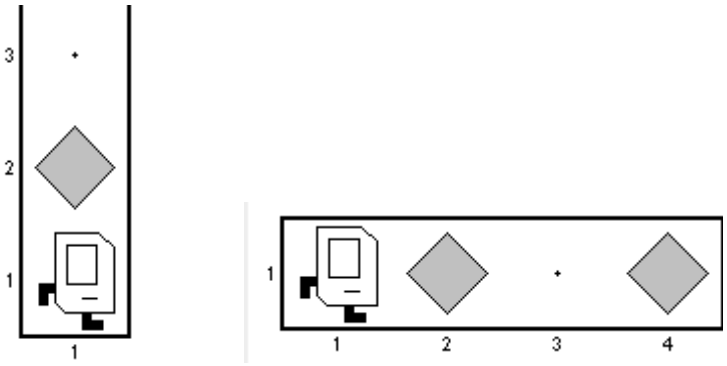
  In this case, Karel has nothing to do, as the map is already one chamber in only three scenarios:

  1x2, 2x1, and 1x1



- ***Two Room Dividing:***

  In this case, there are a limited number of possible maps that can be divided into two chambers, when we have
  2x2, 1x3 or 3x1 and 1x4 or 4x1, since those are all an edge cases and there is no specific algorithm to do the task, I tried to manually use the minimum number of beepers and possibly not the perfect number of moves (since discovering the map is always required), but it logically sounds good for me, as the map is very small here.
  I just put one beeper where needed to get the two chambers map, for example *4x1:* it takes 6 moves to divide(12 moves in total), and two beepers,
  1x3: 2 moves (6 total moves), and one beeper, for mare details, you can check the submitted code.

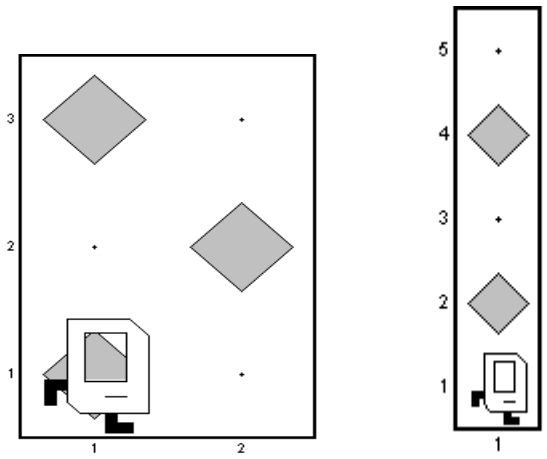


- ***Three Room Dividing:***

  In this case, there are also a limited number of possible maps that can be divided into three chambers, mentioning the six possible cases, *1x5, 1x6, 5x1. 6x1, 2x3 and 3x2,* as the previous cases of dividing into two chambers with the same consideration, I used the minimum number of beepers and moves to achieve this dividing, for example:

  2x3: it takes 12 moves and only 3 beepers

  1x5: 14 moves and two beepers used, note that this can be done by 6 moves but as we said Karel is lost in his world so the move of scanning the map is always counted.



So far, I have solved all the previous cases using Karel's abilities without resorting to mathematical calculations or algorithms. Let's move on to the most exciting part of the problem: four chambers!

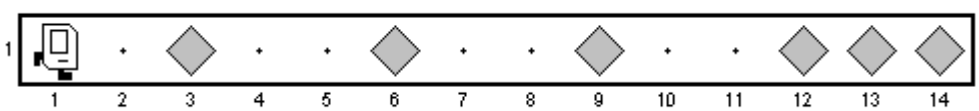
- ***Four Room Dividing***

  This was the most challenging part, as it has a lot of cases to cover, a lot of solutions exist, so I started thinking about how to come up with a good one, with the fewest number of moves and beepers.

  1) *1xN, Nx1 and N >7:*

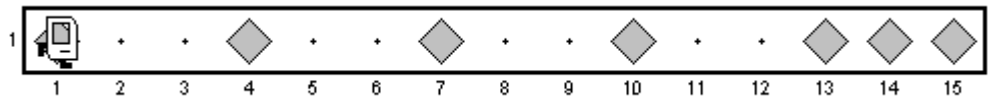
     I will explain the algorithm with 1 height and N width:

     For Nx1, N is even:
     In this case, I used a method to divide any 1xN or Nx1 map with the minimum number of beepers and moves. The technique involved calculating the required chamber size, which can be done by using the formula chamber size = width/4 -1, Karel moves to this position, and he puts one beeper, and then he jumps chamber size +1 to put another beeper, every time Karel puts one beeper, I increase the counter “countChambers” by one, once the countChambers reaches the value of four, I keep adding beepers to the end of the line, for example: *14x1 map:*

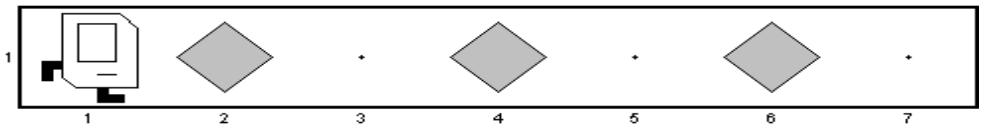


For Nx1, N is odd:

The approach is similar to the previous one when N is even, but with some modifications. I reused the code for even numbers, starting by placing one beeper at the beginning and then moving one step with the new dimension (width - 1), *15x1 map*:

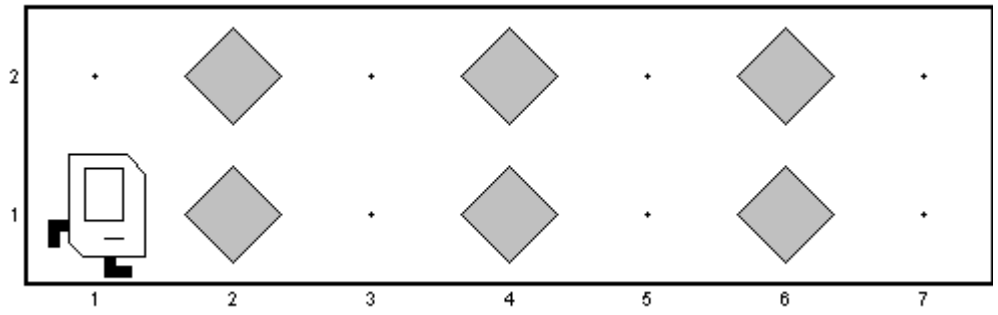


If N=7, I handled this case because the algorithm above does not work with 7, here I just added three beepers using Karel abilities, at the even indeces, it resluts a new map with four chambers with the fewest number of beepers (3 beepers) and moves (10 moves to divide and 22 in total).



2) *For 2x7 and 7x2:*

In this case, I used the same approach as with 1x7 and 7x1, repeating the process twice after moving Karel one step to the second line, it takes 6 beepers and 22 moves to divide, 36 in total



this case was easy, so I just did one more move, and then called an existing method.

In the next cases, I created four methods to handle the division of each dimension:
evenWidthDivider

evenHeightDivider

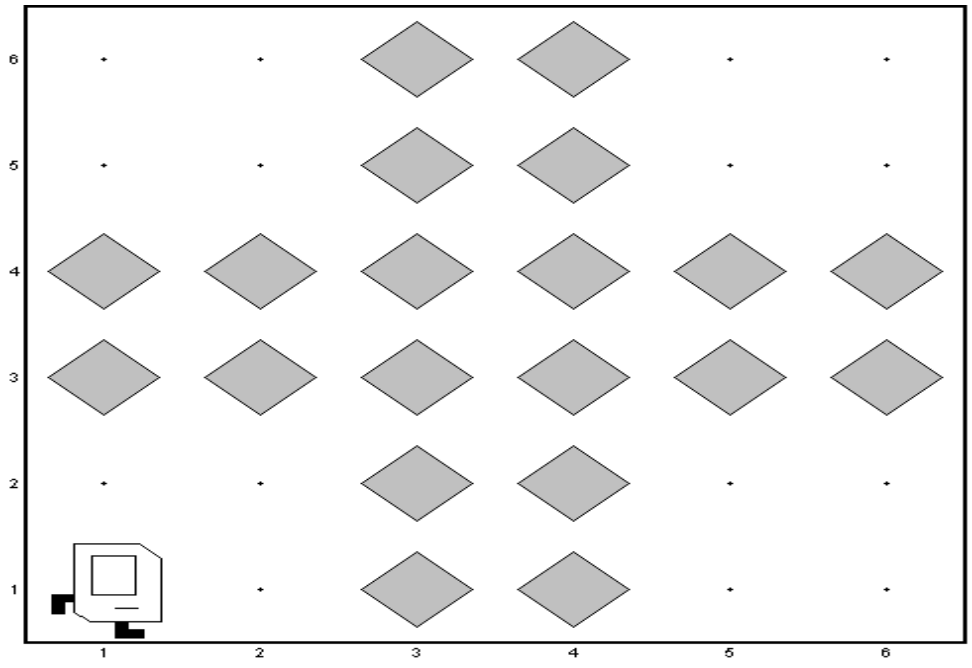
oddWidthDivider

oddHeightDivider

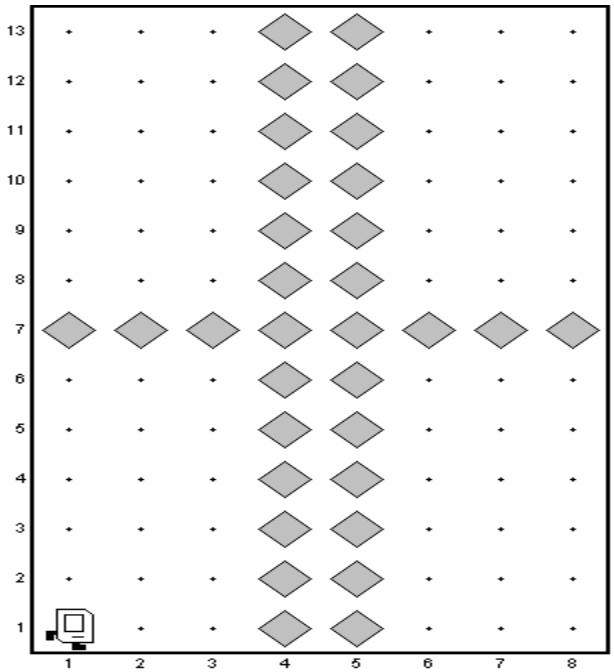
and I have used them when needed.

3) NxM, where N and M are even:

When I have a map with even x even dimensions, I simply call two methods to divide each dimension into two parts, resulting in four parts in total. While this solution may not be the most optimal, due to I puts double line of beepers, but it is the easiest one that came to mind. Which takes more beepers and moves than the optimal one.

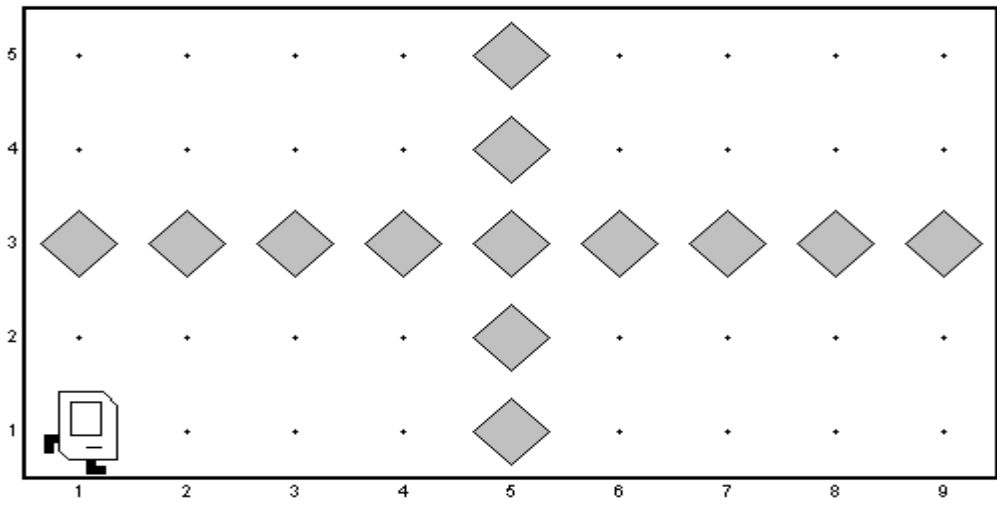


4) NxM, where N is even and M is odd:
for M, it is as easy as you put one line in the M/2, with the minimum number of beepers and moves, and for N, just calling the evenWidthDivider, that will put double line of beepers.



The same for NxM, when N is odd and M is even,

Moving on, when the map is OddXOdd, two lines are enough to divide with the least number of beepers and moves.



You can watch the full video on YouTube following this link: https://youtu.be/IectJI26aUg

Thank You!