Dividing Karel's World into Four Equal Chambers

1. Introduction

This report presents a solution for dividing a given map (Karel's world) into four equal chambers, or the largest possible number of equal chambers if four is not feasible. The solution is optimized to minimize the number of moves and the number of beepers used while also reducing the code length by writing reusable functions.

2. Problem Analysis

The problem requires dividing a grid into equal chambers. The grid may have varying dimensions, and special cases need to be handled when the grid cannot be divided evenly into four chambers. The approach taken ensures that the largest possible number of equal chambers is created, using efficient beeper placement and minimizing Karel's movements.

In some cases, there are multiple ways to divide the grid into equal chambers. To address this, the solution prioritizes methods that first use the minimum number of beepers and second require fewer moves. This dual optimization ensures that the solution is both resource-efficient and time-efficient, meeting the problem requirements effectively.

3. Solution Approach

Step 1: Initialize Karel's Environment

- Set an initial number of beepers (1000) in Karel's bag.

Step 2: Find Grid Dimensions

- Determine the number of columns and rows in the grid using the findKarelWorldDimensions() method.

Step 3: Divide the Grid

- Depending on the dimensions of the grid, divide it into four, three, two, or one chamber(s).

4. Determine how to divide the map

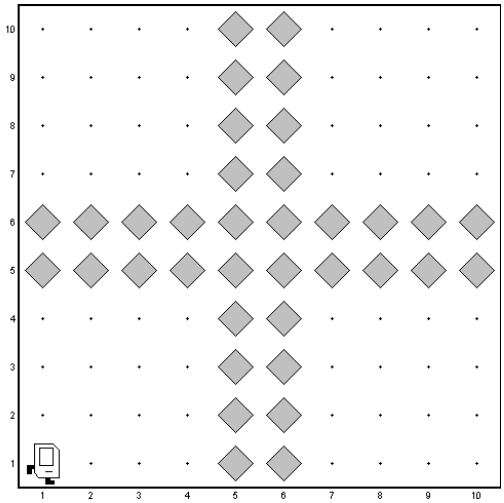
4.1 The dimensions of the map are greater than two

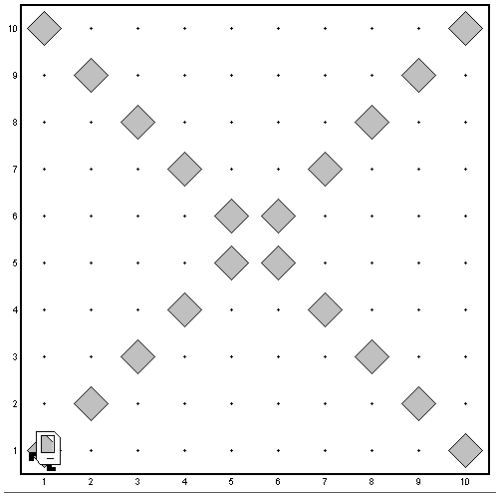
4.1.1 The map is square with even dimensions

There are two ways to divide the map in this case:

1. Double Lines of Beepers (moveDuplicate()): Karel can put double lines of beepers in the middle rows and columns. While effective, this method uses a lot of beepers.
2. Diagonal Division(moveDiagonally()): Karel divides the map diagonally. Although this method requires more moves, it uses fewer beepers and creates the largest possible chambers.

Due to its efficiency in beeper usage, the diagonal division is chosen despite the higher number of moves.



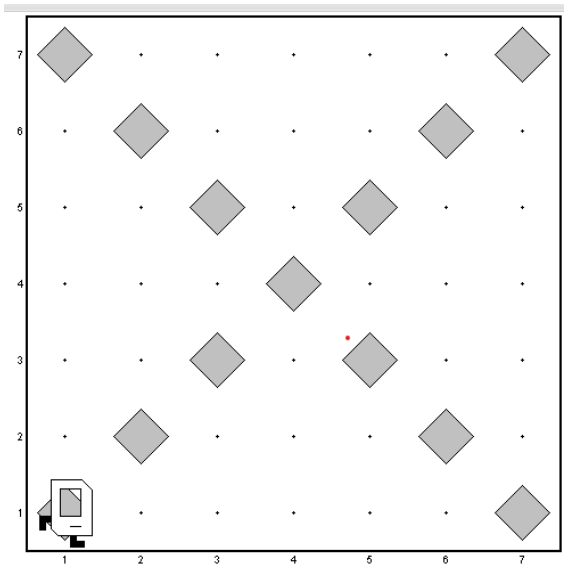


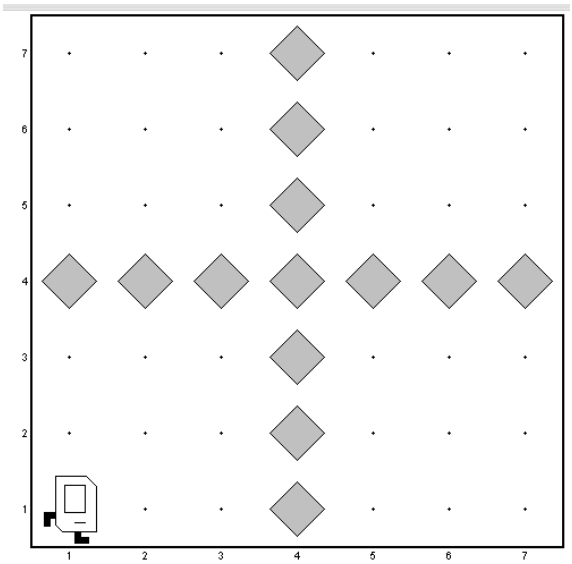
4.1.2 The map is square with odd dimensions

There are two ways to divide the map in this case:

1. Diagonal Division(moveDiagonally()) .
2. Cross Division (moveAsPlusSign()) : Karel will make a line of beepers in the middle row and in the middle columns and will use the same number of beepers.

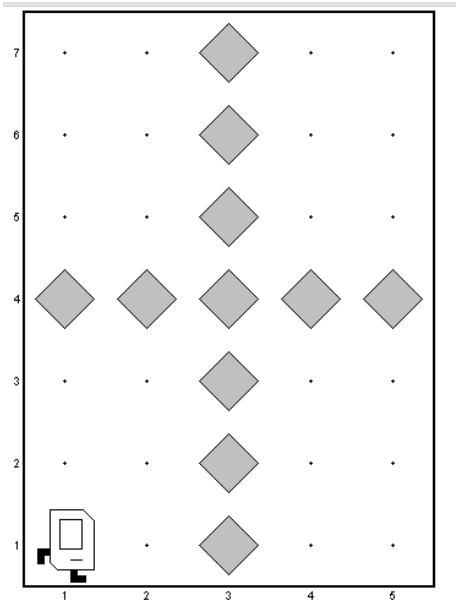
   Both methods use an equal number of beepers, but the cross division (moveAsPlusSign()) is chosen for its efficiency in movement, as it achieves equal chambers without requiring any additional moves beyond the placement of beepers.





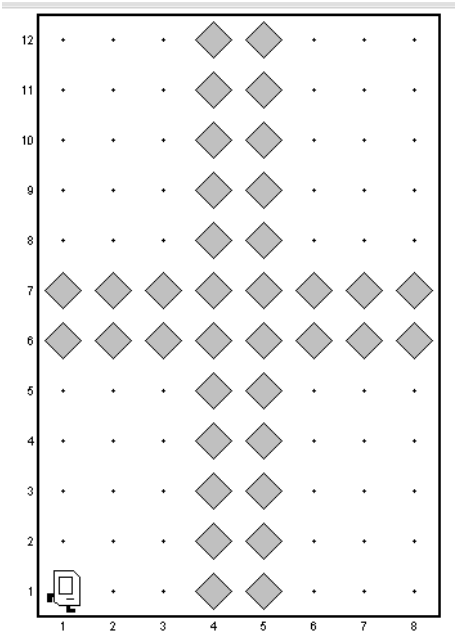
4.1.3 The map is rectangle with odd dimensions

Cross Division (moveAsPlusSign()).



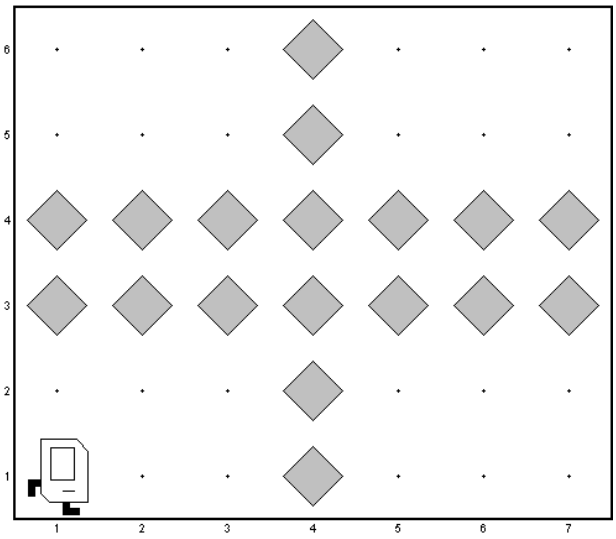
4.1.4 The map is rectangle with even dimensions

Double Lines of Beepers (moveDuplicate()): Karel can put double lines of beepers in the middle rows and columns. This method uses a lot of beepers but in number of moves is optimized.



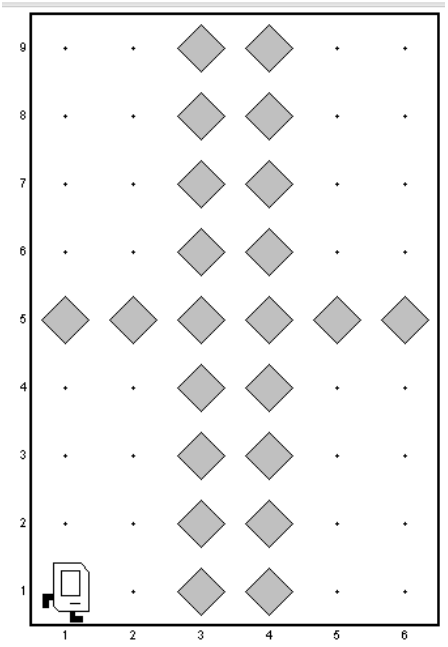
4.1.5 The map is rectangle with even rows and odd columns

Two line of beepers (dublicateRows()) :Karel will put two lines of beepers in the middle rows and one line of beepers in the middle column.This method uses a lot of beepers but in number of moves is optimized.



4.1.6 The map is rectangle with even columns and odd roes

Two line of beepers (dublicateColumns()) :Karel will put two lines of beepers in the middle columns and one line of beepers in the middle row. This method uses a lot of beepers but in number of moves is optimized.



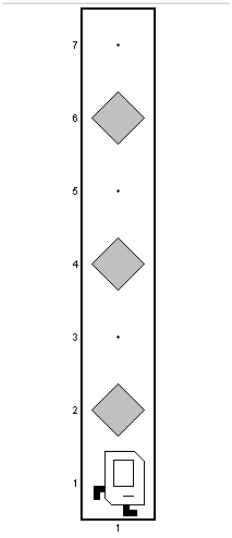
4.2 One of the dimensions of the map is two or one and the other is greater than six

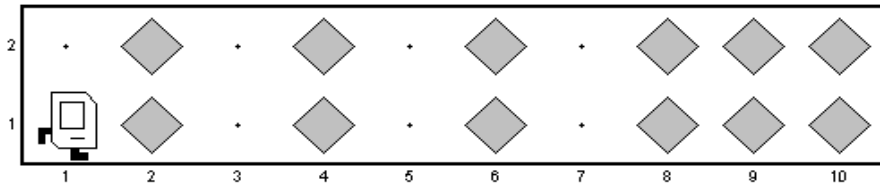
step 1 : Determine how many node will be in one chamber using (noOfNodesPerChamber()) method :

1. Determine the max value between rows and cols.
2. Suppose the number of nodes per chamber is the max divided by four. If the result is equal to or greater than 3 (because three lines are needed to divide and place the beepers), return the result. Otherwise, increment the number of nodes per chamber until the result becomes 3 or greater.

```
public int noOfNodesPerChamber() {   //>=7
    int m = Math.max(rows, cols);
    int noOfNode = m / 4;
    if ((m - (noOfNode * 4)) >= 3) {
        return noOfNode;}
    while (noOfNode > 0) {
        noOfNode--;
        if (m - (noOfNode * 4) >= 3) {
            return noOfNode;}}
    return 0; }
```

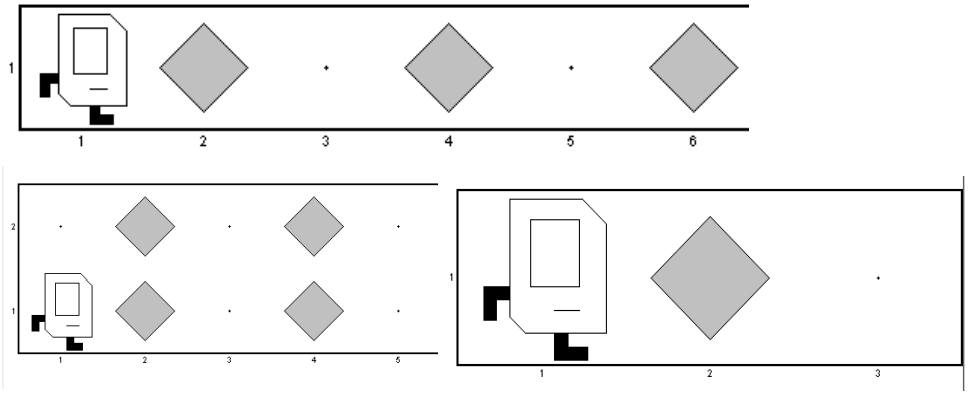
Step 2: Divide using (divideAsPatterns()) method which will make Karel divide after certain number of nodes determine by (noOfNodesPerChamber()) method and place beepers in the remaining areas. In addition when rows or columns are equal to two we will call (divideAsPatterns()) two times.





4.3 One of the dimensions of the map is two or one and the other is less than six

using (twoOrThreeChambers()) method which will determine the max value between rows and cols. Then divide to two or three chambers with one node in each chambers. . In addition when rows or columns are equal to two we will call (twoOrThreeChambers()) two times.



5. Optimization Details

1. Minimized Moves: The moveToEnd and findKarelWorldDimensions methods ensure minimal moves by utilizing efficient traversal strategies.

2. Reusable Functions: Functions like moveToEnd, findKarelWorldDimensions, divideIntoFourChambers, and twoOrThreeChambers are designed to be reusable, reducing code duplication and increasing readability.
3. Efficient Beeper Usage: Beepers are placed only where necessary to divide the chambers, optimizing their use.

6. **Enhancement : Efficient Grid Resizing and Navigation for Karel**

   In our ongoing effort to improve Karel's efficiency and adaptability in navigating and manipulating grid environments, we have implemented a grid resizing enhancement. This enhancement is particularly useful when Karel needs to operate on grids where the difference between the number of rows and columns is less than 3. The resizing method ensures the dimensions of the grid are odd, facilitating easier division using the plus sign method, which is efficient in terms of both beepers and movements. Additionally, when there are remaining parts of the map with 2 or 1 columns or rows, Karel fills the remaining area with walls to resize the map.

7. **Conclusion**

   The solution effectively divides Karel's world into four chambers or the maximum possible number of equal chambers if four is not feasible. The implementation is optimized for the number of moves and beepers used, providing an efficient and concise solution to the problem.