Karel Assignment Report

By Hamza Alhalabi

Contents

Problem Statement

Dividing a map into 4 equal parts, if dividing into 4 equal parts is not possible, divide into the largest possible number (3, 2, or 1).

Approach

Initial Solution

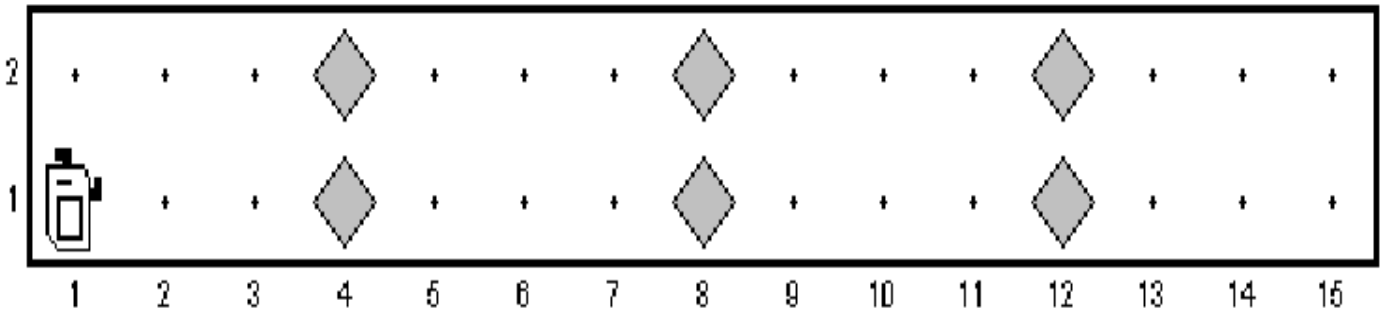
- At first, I divided the logical cases as follows:
  i. If the map grid was (1x1) (2x1) or (2x2) I considered it undividable (**later optimized the division of (2x2) using diagonal lines**).
  ii. If one dimension is 2 or less, and the other dimension is 3 and above, I divided it with specific logic: If the larger dimension subtracted by 3 is divisable by 4, that means I can divide the map into 4 chambers. Otherwise I test if the larger dimension subtracted by 2 is divisable by 3, it is possible to divide into 3 chambers. if the two previous conditions didn't apply, I divide the grid into 2 equal chambers.

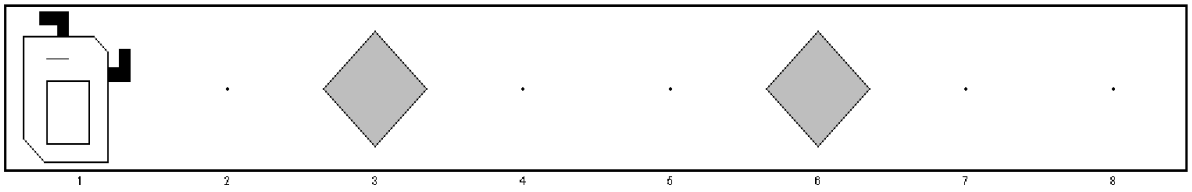
iii. When both dimesnions are 3 or more, I divided the map with different logic: The odd dimension is divided by a single line of Beepers, and the even dimension is divided by a double line of Beepers (**I added some optimizations later**).

- After determining the algorithm, I started writing the code. At first the robot calculates the width and height of the map, then it chooses the appropriate way to divide that map by a conditional structured statements and start putting Beepers as needed.
- The program prints to the console the numbers of moves while the robot's movement. With the end of the program, it prints 4 statements that show the dimensions of the map, the total number of moves, the total number of used Beepers, and a statement that appears only if the division is not possible.

Challenges Faced

- I was worried about the idea of using **diagonal** lines of Beepers. At the end, I used diagonals with some optimizations only.
- I faced some ambiguity while building Beepers' line logic in the case of (**2** or **1** x **3** or **above**). I felt like there are many ways (both complicated and simple) to solve this. But finally I chose a logic that prioritize using less Beepers over creating larger chambers with complicated shapes.

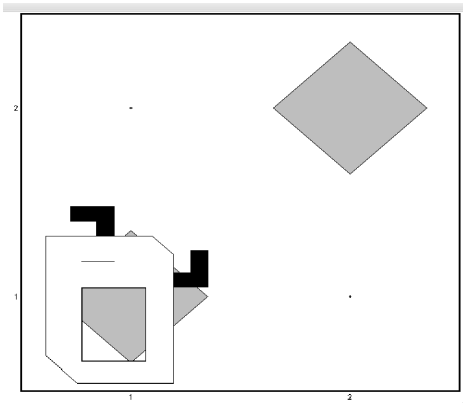




- When I began with the assignment, there was an error related to configurations, the SDK version, and compatibility with the *Stanford.Karel* library that took me 2 days to fix.

Optimizations

- I optimized the dividion in the case where both dimensions are the same and even, drawing diagonal lines of Beepers is better for less moves and less Beepers usage, this only worked for even equal dimensions not with odd equal dimesnions.
- Although I tried to avoid the complexity of diagonals when one dimension is 2 or less, I decided to draw a single diagonal line for a (2x2) map.



- I optimized the lines of code as much as possible. I ensured to avoid redundant lines of code and aggregated similar functionalities into one method to reuse for different cases.
- I wrapped up some inherited methods like *move()* and *putBeeper()* to add some operations and conditions:

```
private void moveCounted(){
    move();
    movesCounter++;
    System.out.println("Current moves: " + movesCounter);
}

private void putBeeperCounted(){
    if(noBeepersPresent()){
        putBeeper();
        beepersCounter++;
    }
}
```

- Before optimizing the number of moves, the robot was returning to a near corner after putting all Beepers, this added unnecessary moves equivalent to half the height of the map. But I optimized that so the robot knows that this is the last line of Beepers and there is no need for moving to another corner.