

## Map Division Report

# ATYPON

**Author:** Ahmad Emad

**YouTube Video:** [Map Division Demonstration](#)

**Email:** [ahmademad995.ae@gmail.com](mailto:ahmademad995.ae@gmail.com)

### Objectives:

- Divide any given map into four equal chambers.
- If four chambers are not possible, divide the map into the highest number of chambers achievable.
- Minimize the number of code lines.
- Reduce the number of moves required.
- Use the smallest number of beepers possible.

## Step-by-Step Solution:

1. Calculate the map dimensions (x-axis and y-axis lengths).
2. Categorize the axes and call the appropriate method based on
  - **Odd Axes**
  - **Even Axes**
  - **Odd-Even Axes**
3. Further analyze the axes to call appropriate division methods and handle indivisible axes

## Main:

```
public static int moveCounter = 0, beeperCount = 0; 1 usage

public void run() { Ahmad Emad *
    startPosition();
    int[] xy = calculateArea();
    int x = xy[0], y = xy[1];

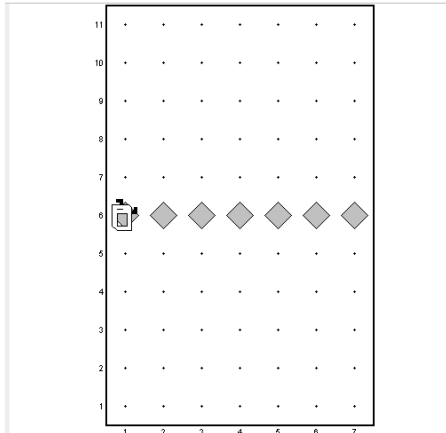
    if (x % 2 == 1 && y % 2 == 1) oddAxes(x, y);
    else if (x % 2 == 0 && y % 2 == 0) evenAxes(x, y);
    else oddEvenAxes(x, y);

    println("Beeper count = " + beeperCount);
}
```

## Division Methods:

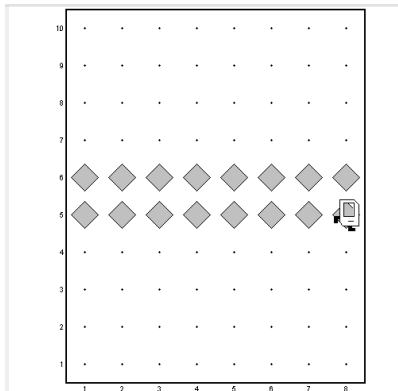
- **divideY(int y)**

Divides the y-axis into two equal halves with one wall of beepers. Suitable for odd y-axis lengths.



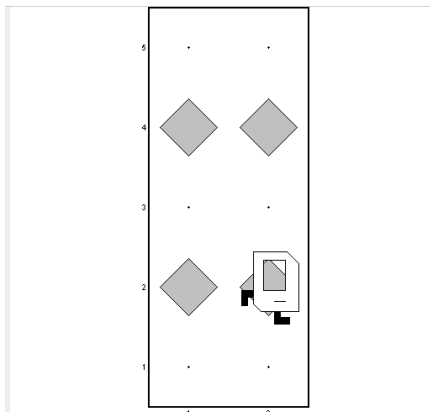
- **doubleDivideY(int y)**

Divides the y-axis into two equal halves with two walls of beepers. Suitable for even y-axis lengths.



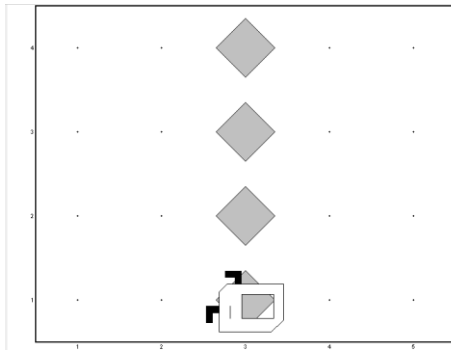
- **threeChambersY(int y)**

Divides the y-axis into three parts using two walls of beepers. Suitable for y-axis lengths divisible by 3, especially when x-axis  $\leq 2$ .



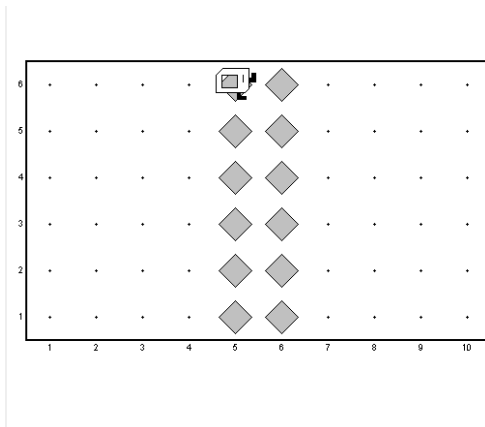
- **divideX(int x)**

Divides the x-axis into two equal halves with one wall of beepers. Suitable for odd x-axis lengths.



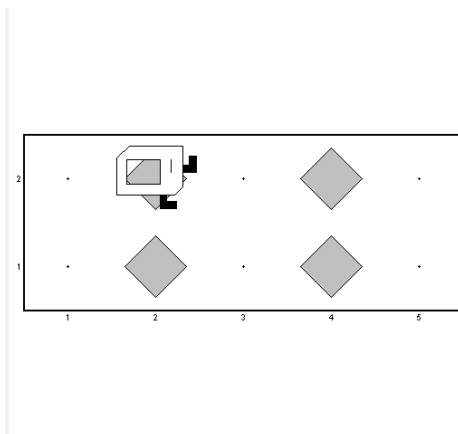
- **doubleDivideX(int x)**

Divides the x-axis into two equal halves with two walls of beepers. Suitable for even x-axis lengths.



- **threeChambersX(int x)**

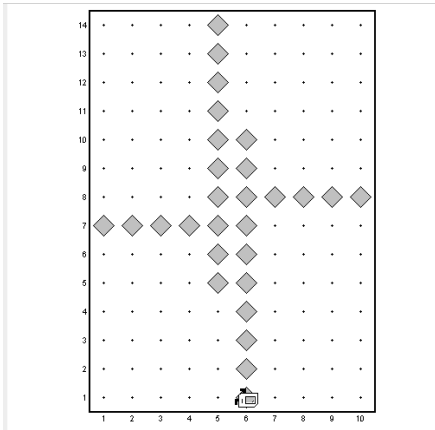
Divides the x-axis into three parts using two walls of beepers. Suitable for x-axis lengths divisible by 3, especially when y-axis  $\leq 2$ .



## Curving Methods:

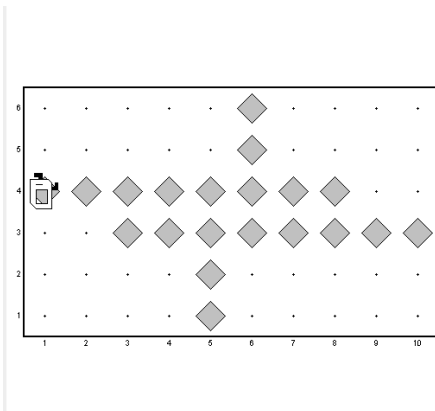
- **yAxisCurve(int x, int y)**

Creates an overlapping curved vertical wall of beepers to divide the map into four chambers. Suitable when both axes are even, and y-axis is taller.



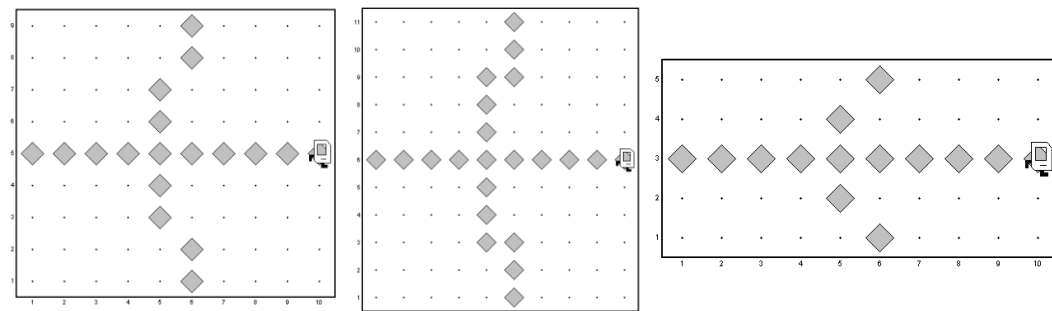
- **xAxisCurve(int x, int y)**

Creates an overlapping curved horizontal wall of beepers to divide the map into four chambers. Suitable when both axes are even, and x-axis is wider.



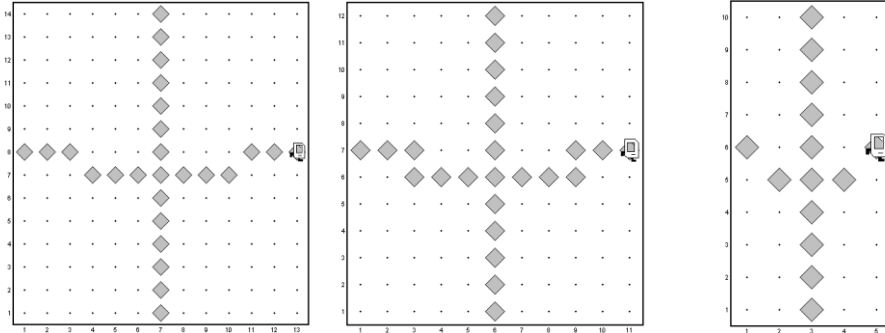
- **verticalCurve(int x, int y)**

Dynamically adjusts to create a curved vertical wall of beepers. Suitable for even x-axis lengths and odd y-axis lengths.



- **horizontalCurve(int x, int y)**

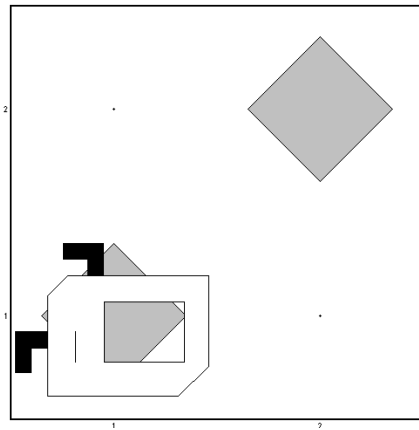
Dynamically adjusts to create a curved horizontal wall of beepers. Suitable for even y-axis lengths and odd x-axis lengths.



## Special Case Method:

- **twoByTwo()**

Handles the specific case where both the x-axis and y-axis lengths are 2, dividing the map diagonally into two chambers.



## Analysis Helper Methods:

- **oddAxes(int x, int y)**  
Determines the best method for dividing the map when both axes are odd, such as using threeChambersX() or divideY() and divideX().
- **evenAxes(int x, int y)**  
Determines the best method for dividing the map when both axes are even, such as using yAxisCurve() or xAxisCurve().
- **oddEven(int x, int y)**  
Determines the best method for dividing the map when one axis is odd and the other is even, such as using verticalCurve() and divideY().
- **calculateArea()**  
Calculates the lengths of the x-axis and y-axis.

## Reusable Helper Methods:

- **moveWhileFrontClearPutBeeper()**  
Repeatedly moves forward and places a beeper while the path ahead is clear.
- **moveBy(int steps)**  
Moves a specified number of steps forward.

## Overridden Methods:

- **move()**  
Tracks the number of moves made by incrementing a move counter every time this method is called.
- **putBeeper()**  
Checks if no beeper is present before placing one, preventing overlap. Also increments a beeper counter to track usage.

## Conclusion:

This solution employs a **top-down approach**, starting by categorizing the map axes into three main categories: even-even, odd-odd, and odd-even. Based on the category, it determines the appropriate methods to divide the map.

## Optimization:

- **Less Redundancy:** Overridden methods to exclude repetitive condition checks while also tracking moves and beepers. Used reusable methods e.g., `moveWhileFrontClearPutBeeper()` which has 16 usages, and `moveBy()` which has 13 usages promoting clean and efficient code.
- **Less Beeper Use:** Curving methods like `yAxisCurve()` and `xAxisCurve()` reduce the number of beepers by overlapping walls where possible.
- **Higher chambers:** The map is ensured to be divided into four equal chambers or the next possible highest number of chambers.

At only **277 lines of code**, this solution effectively balances functionality, efficiency, and clarity.