# Karel Assignment

## Atypon/Wiley

Hadeel Abunassar

For dividing the map into 4 equal champers or the biggest possible number of equal chambers [3,2,1]  I had to handle too many cases , but before presenting them I am going to explain some main methods that I used continuously in my code.

- ### getDem() method

  This method main and only purpose is to scan the map to get the dimensions, this method is invoked always before the dividing process. The robot will walk through the x-axis and count the number of moves, then the same thing for the y-axis . At the end the robot will return to the origin point.

- ### countMove() method

  Since we are required to count the number of steps , I wrote a countMove method that I used instead of move() method . this method will add 1 to a global variable called count after each invocation for the move() method.

- ### putAllBeepers(int x) method

  This method will put beepers along 0 position to the Xth position.

- ### moveDiagonal (int x) method

  This method will move in the map diagonal and put beepers along it, its only used when I want to divide square map with even dimensions.

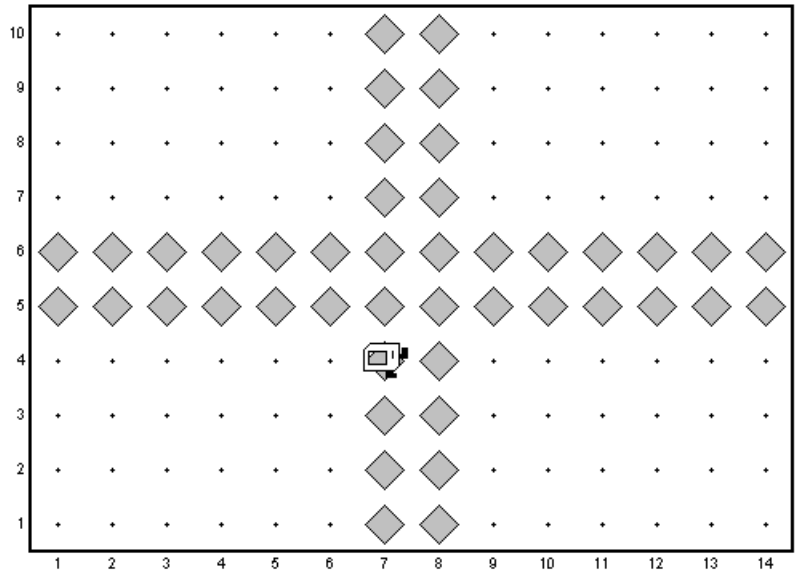
- ### moveHalfMap (int x) method

  Since in my dividing methods the robot will move to the middle of the axis a lot , I had to implement a method to serve this purpose.

In the Run -main- method, after the invocation of the getDim() method,I invoked divideMap(int x, int y method to start the division process. The implementation of this method depends mainly on the map dimensions x and y based on this criteria:

1- if both X and Y are bigger than 3 :

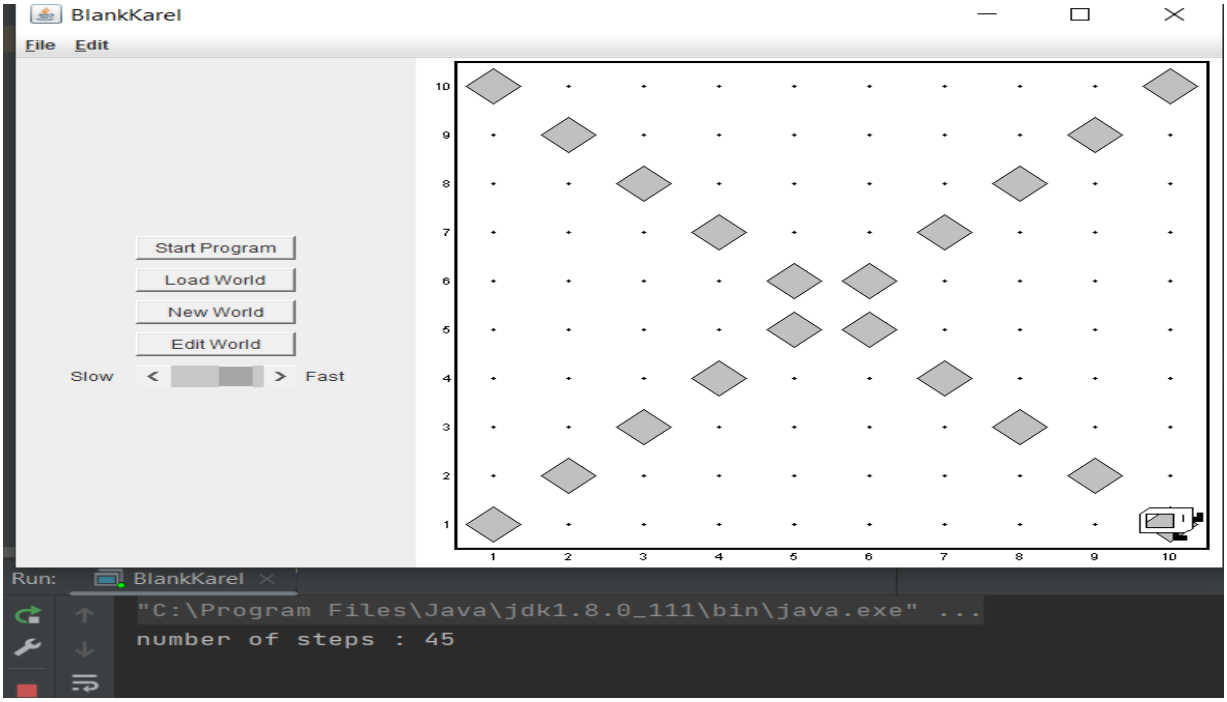
- if X & Y are both even EvenMaps method will be invoked , this method will put double walls in both axes in this way :



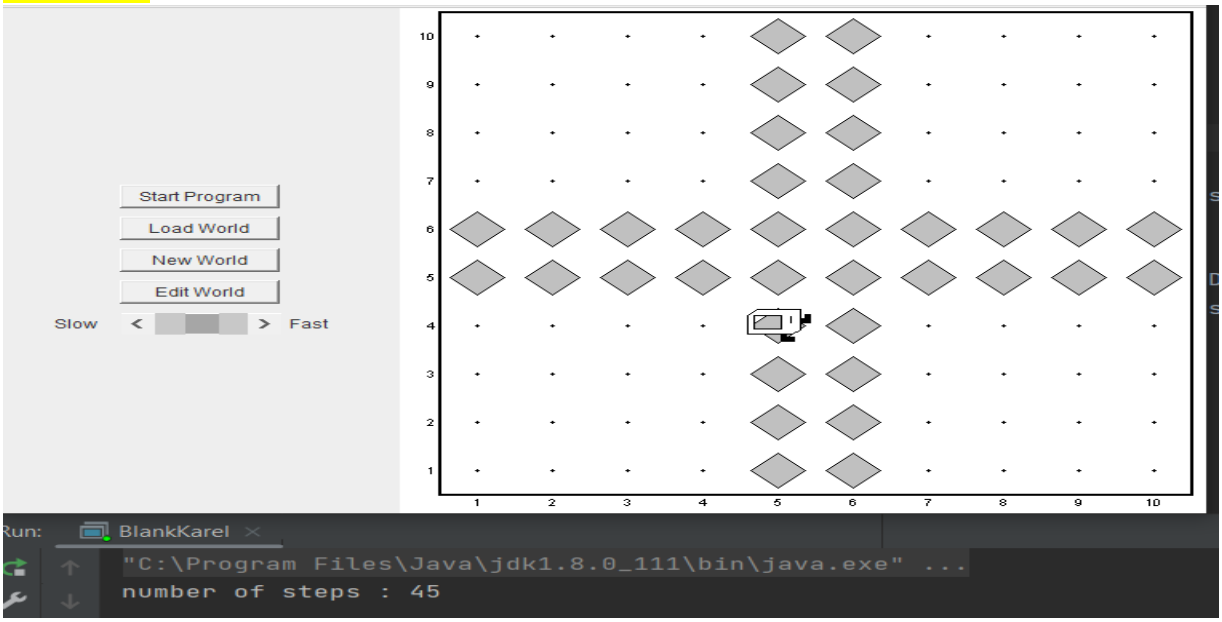

For optimization purposes , after putting double wall in the middle of the y-axis , I will go back to the middle using the already created wall , going up to (x/2,y) then using putAllBeepers method to put beepers along (x/2)+1.

I tried first to make the robot walk through (1,(y/2)+1) to (1,(y)) then going to ((x/2),y). But the second approach reduced the steps by –(x/2)

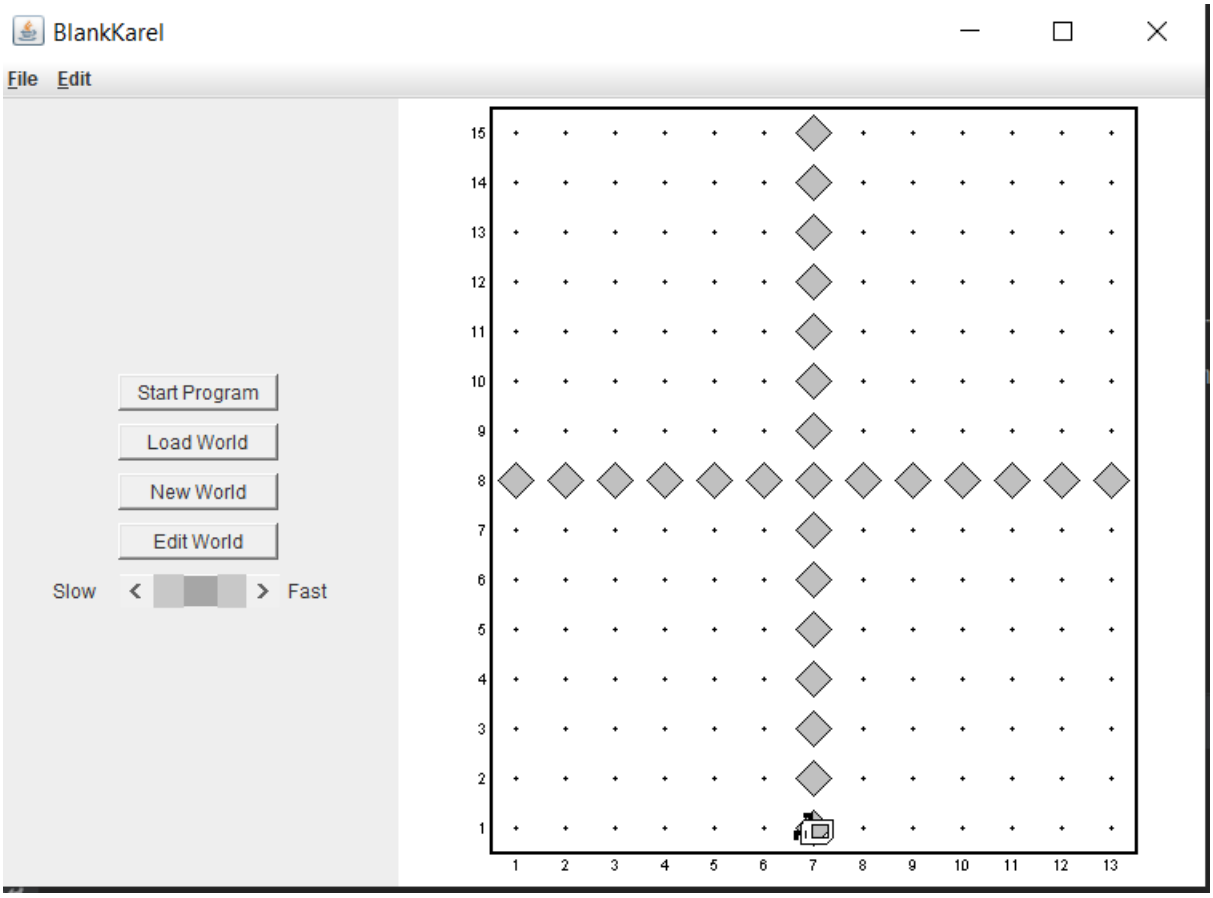
- if X & Y are both even and EQUAL.
  In this method I am going to put beepers along the diagonals



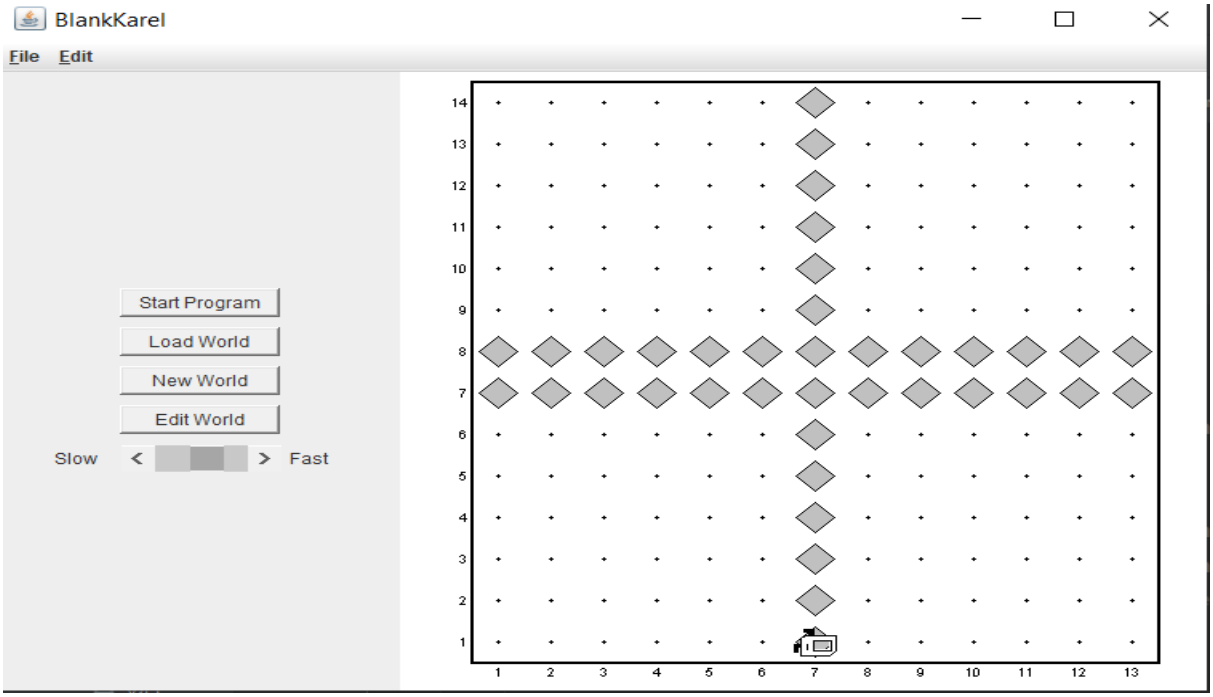

Actually , this is a special case for the even maps , If we tried to invoke the first method on a square map we will get the same numbers of steps , but the Second approach were we put beepers along the diagonal is optimized version because we will use less amount of beepers.



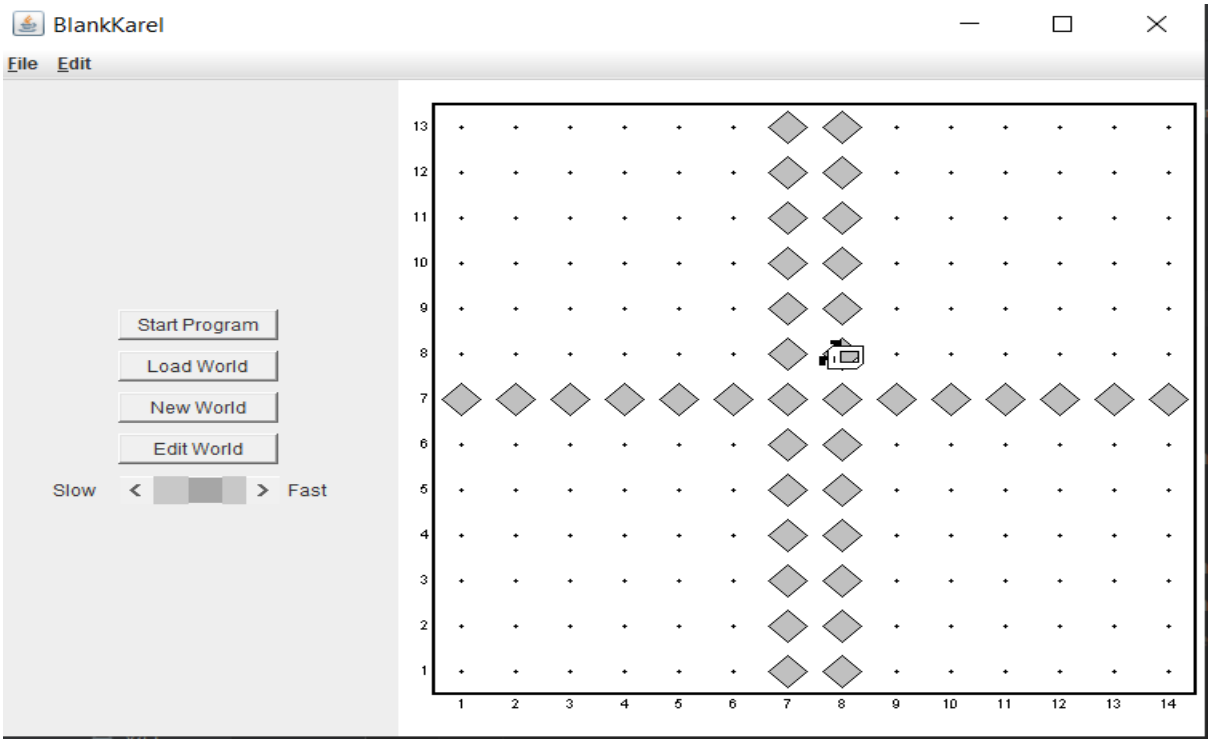
- If X & Y are odd :
  The robot will Put beepers in the middle of the both axes



- If x is odd ,Y is even .. or vice versa
  If Y is even:




If X is even: and in this I used the same optimization approach in the EvenMaps. Where I reduced the number of steps by making the robot going back to walk through the created wall.

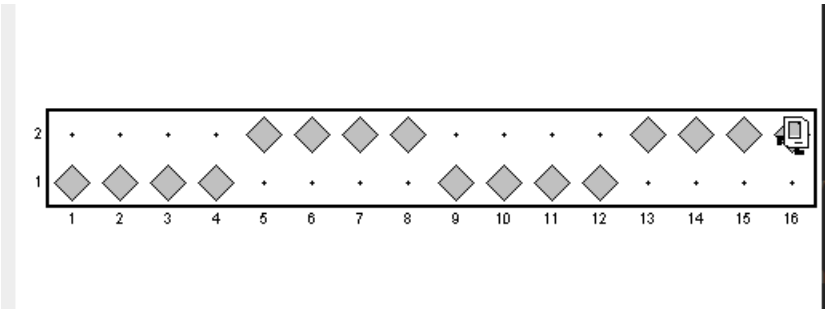


2- if X or Y equals 2 :

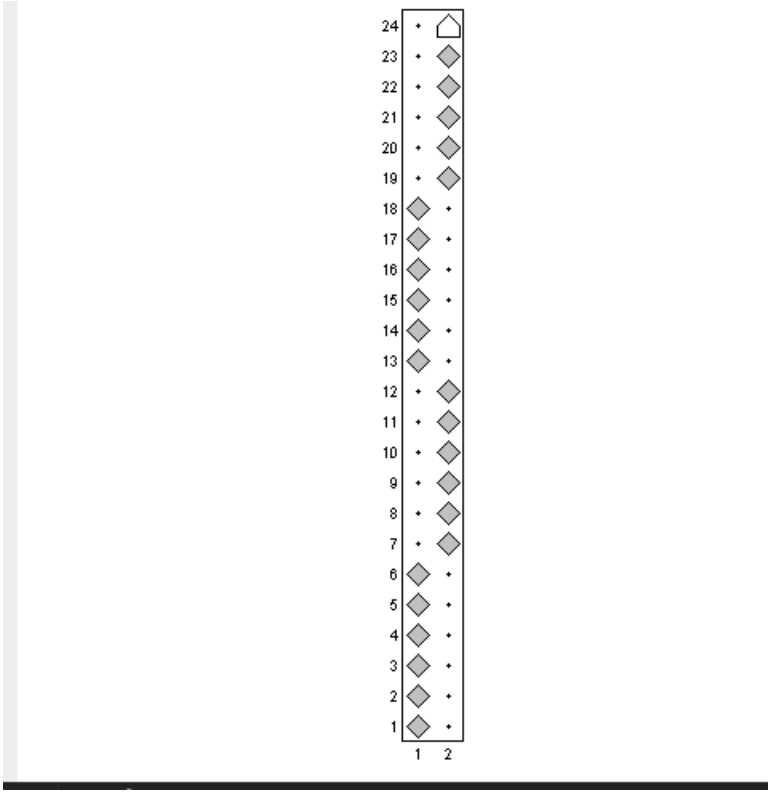
For this case , I had implement a method called hasDimTwo. This method has 3 main cases: first is when the !2 dimension is odd , second is when the !2 dimension is even and has an even remaining when I divide the dimension by 2 , and the third and last case when I have an odd remaining.

I considered the second case is the main case , and the 1st and 3rd are special cases from the main case

For the remaining == even :

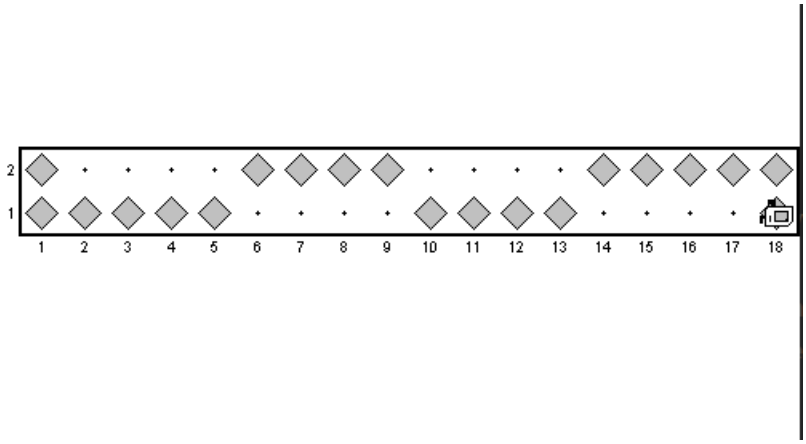



It will always iterate four times and put (!2 dimension /4) beepers in each iteration,
For this case we put (16/4) four times .



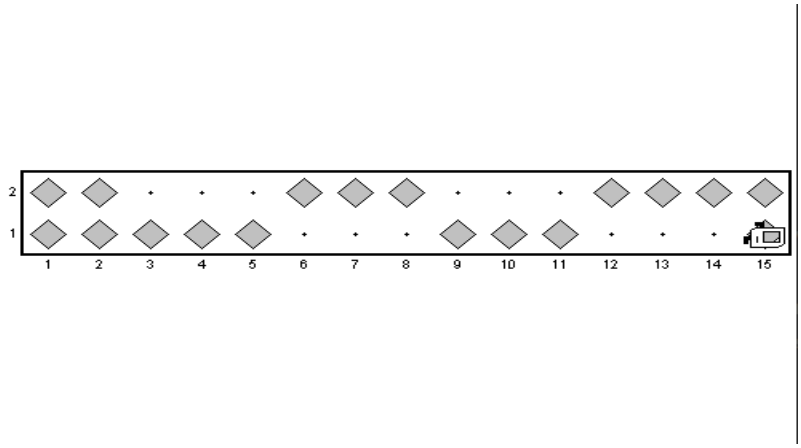

Here we can see that we put (24/4) beepers four times .

For remaining == odd :

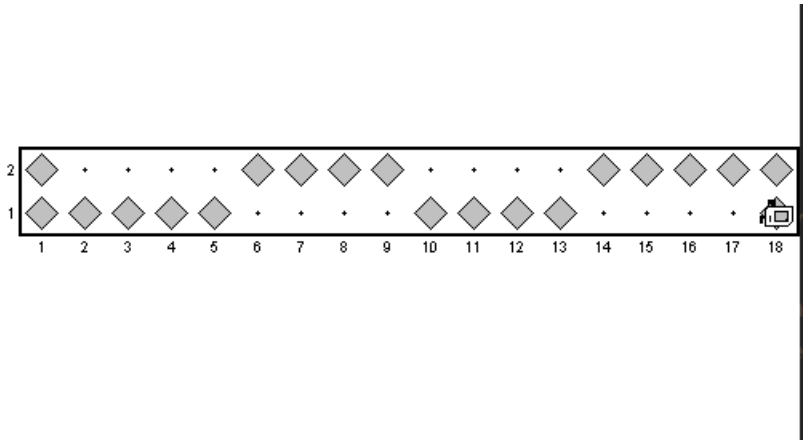



We can see that we have followed the same previous approach but I put two parallel walls to the edges , this made the map as it was like (2*16) which make us go back to the remaining == even

For the !2 dimension == odd
the robot will put a single wall parallel to the edge near the origin position, which will convert the dimension to an even one , so we will then decide which division approach to use depending on the remaining

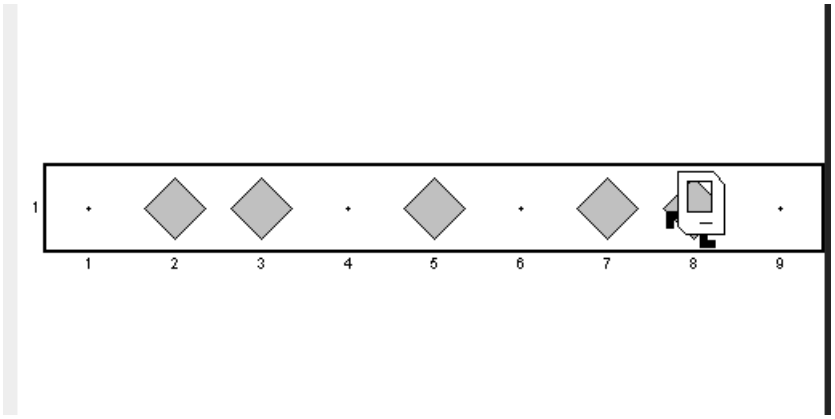



15-1 = 14%2 = 1 (odd)

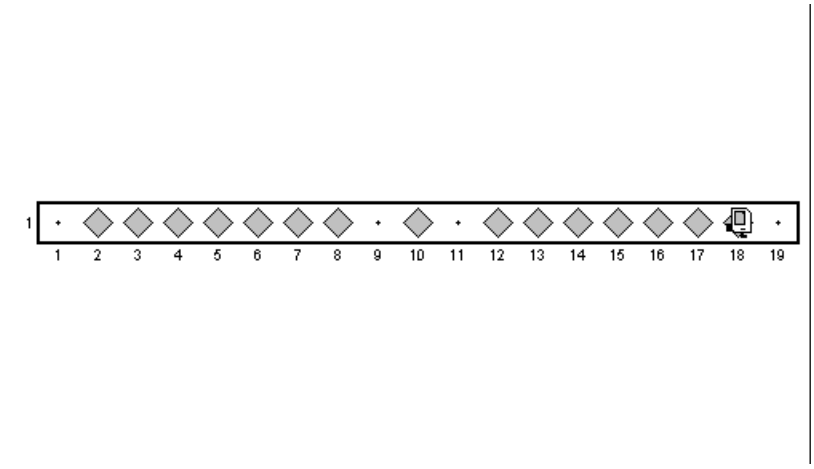



13-1 = 12%2 = 0 (even)

3- if I have one dimension = 1 :
if the other dimension is odd :
x = 9

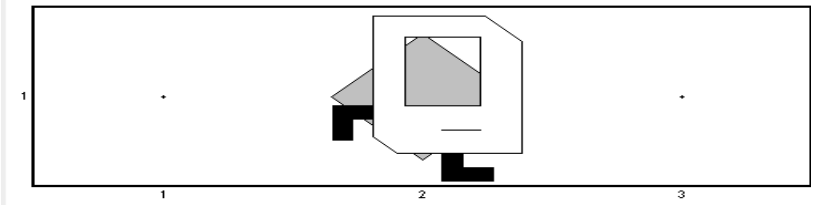


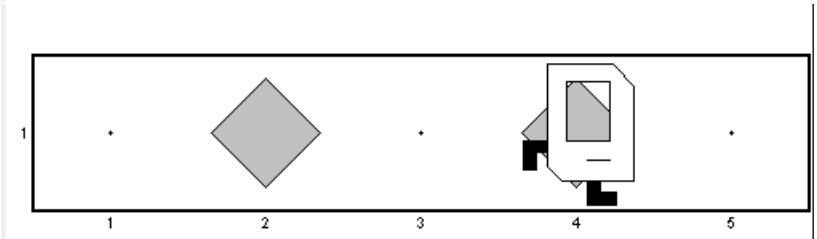

X = 19



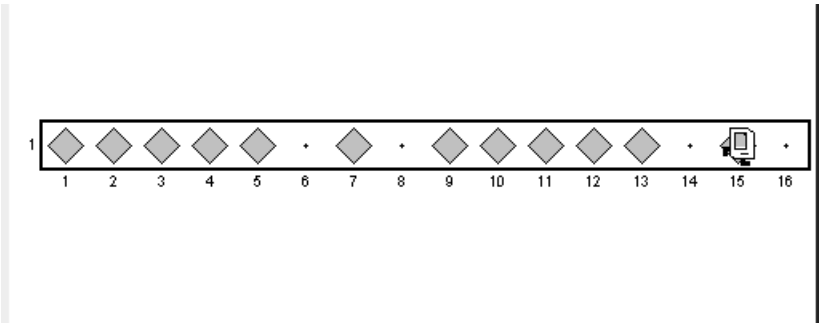

In this case , there will always be a single beeper in the middle of the map . Regrading the number of beepers near the both edges it will be = (x-5)/2 or (y-5)/2So,
In the first example X = 9 . So, the number of steps will be (9-5)/2 which equal 2
In the second example (19-5)/2 which is equal 14/2 = 7.

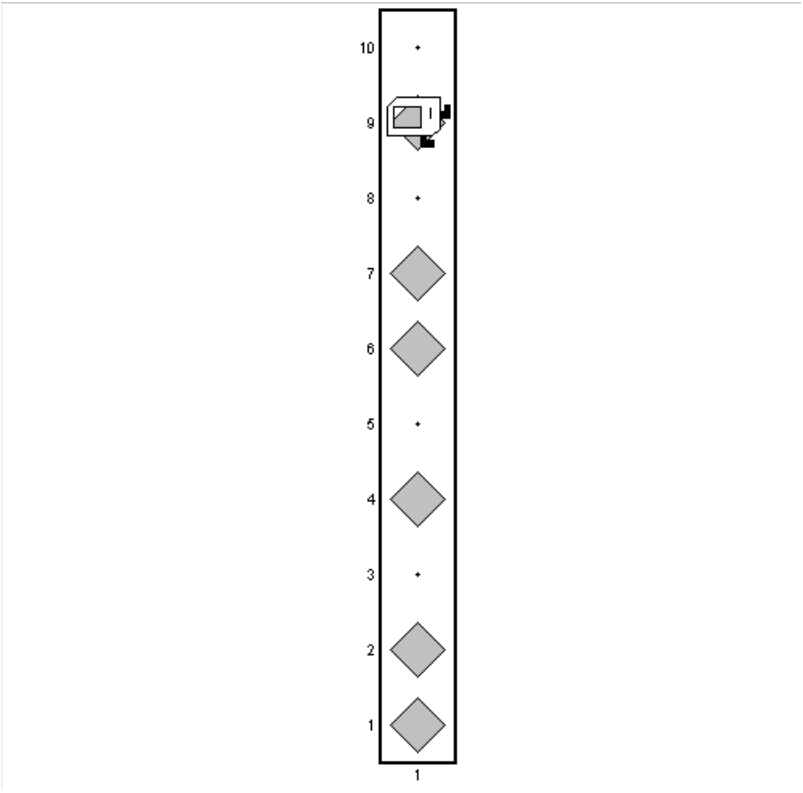
This case has two corner cases that I had to hardcode, when x/y = 3/5 .



- one dimension is one and the other is even



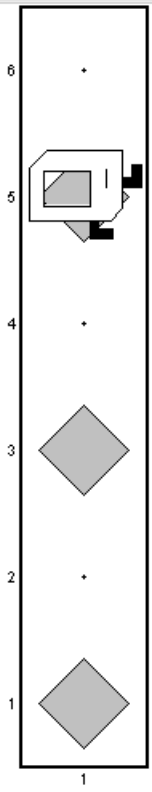

Just like the previous case, number of beepers in the first & third iterations will be:
(x-6)/2) or (y-6)/2)

In the first example, I have x=16 -> (16-6)/2 = 5
In the second example , I have y=10 -> (10-6)/2 =2

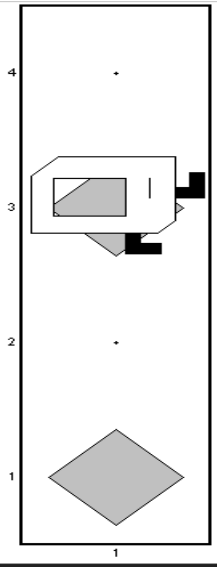
Starting form the origin will reduce the number of steps by 1 because the robot won't reach the end.

Just like the previous case, I had 2 corners cases that I had to hardcode: when x/y=4/6.

Y=6:




Y= 4 :



Finally, the only remaining case that we can divide the map to equal chunks is when x=2 & y=2 :