Karel Assignment Solution Report

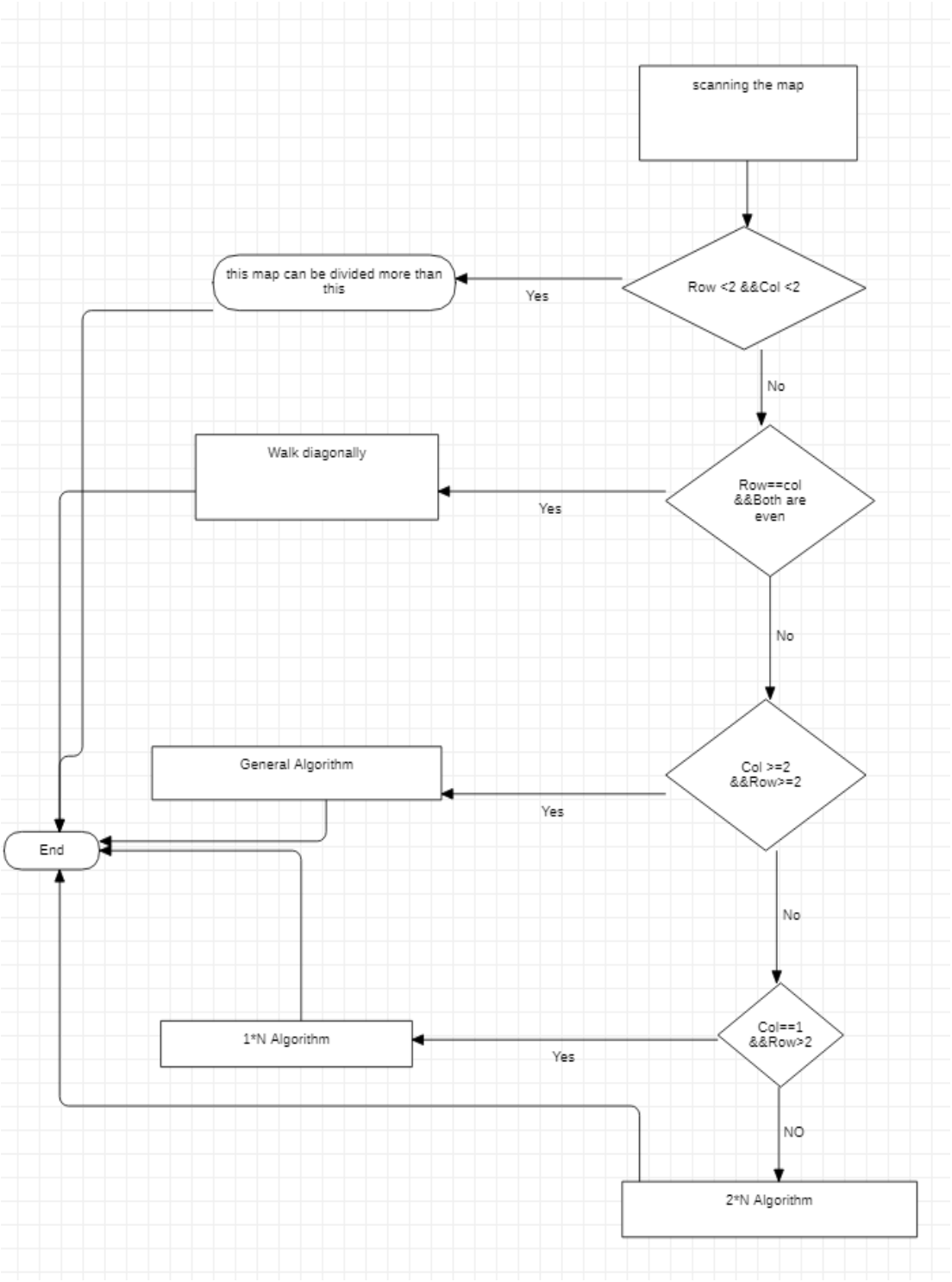
Mahmoud Haifawi

Abstract:

The problem was to create an optimized algorithms in Java to automate Karel to divide any possible grid into four equal areas.

The problem has multiple cases, so I wrote four algorithms to ensure my solution covers any map with the least number of beepers and steps possible.

Flow chart :


scanning the map
Row <2 &&Col <2
Yes
this map can be divided more than this
No
Row==col &&Both are even
Yes
Walk diagonally
No
Col >=2 &&Row>=2
Yes
General Algorithm
No
Col==1 &&Row>2
Yes
1*N Algorithm
NO
2*N Algorithm
End

Flow Chart Summary:

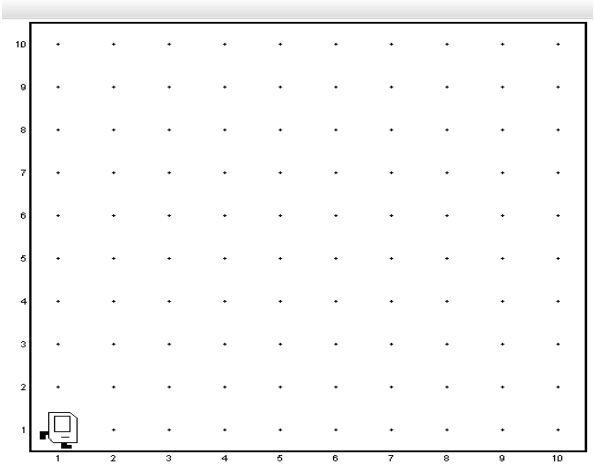
The Karle will scan the map, to determine the number of row and columns and print the results , then Karel will be smart enough to choose which algorithm to use depending on these numbers if the number of rows and columns are less than 2 , the program will terminate, otherwise the program will check if the numbers are (equals and even) or not, if the condition is true then the Karel will divide the map using the diagonal algorithm ,otherwise it will enter another if statement to check if both numbers are bigger than 2 , if the returned value is true, the Karel will apply a general algorithm that is passed by 90% of the test cases , otherwise the Karel will decide if the (column equal 1 and rows > 2 or versa), if the condition was true the Karel will apply 1*N algorithm otherwise will apply 2*N algorithm.

Algorithms description:

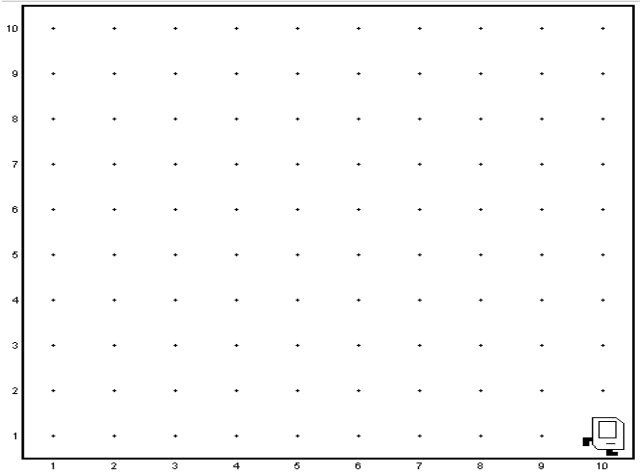
1) Scanning Algorithm:

The scanning method is been called only once, when we started a new run, the program will rest the counters of(row, columns, steps, beepers) to make the program run multiple times whenever we load a new map and run it without the need to terminate the program to start a new run.

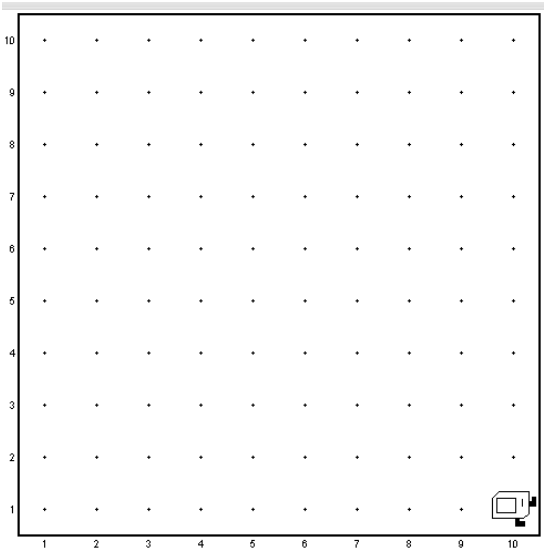
The starting point for the Karel is (0,0)



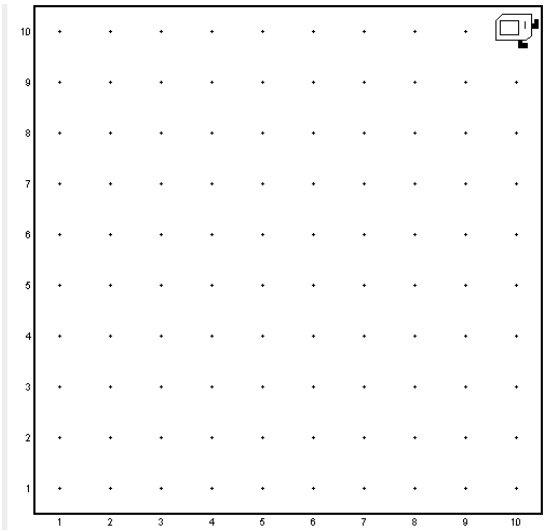
the Karel will move horizontally while the front is not blocked by the wall, to count the columns numbers, and by every step it will increase the columns and steps counter



When the Karel found the front is blocked it will turn left using the built-in method



Then it will move vertically while the front is not blocked by the wall, to count the rows numbers, and by every step it will increase the columns and steps counter

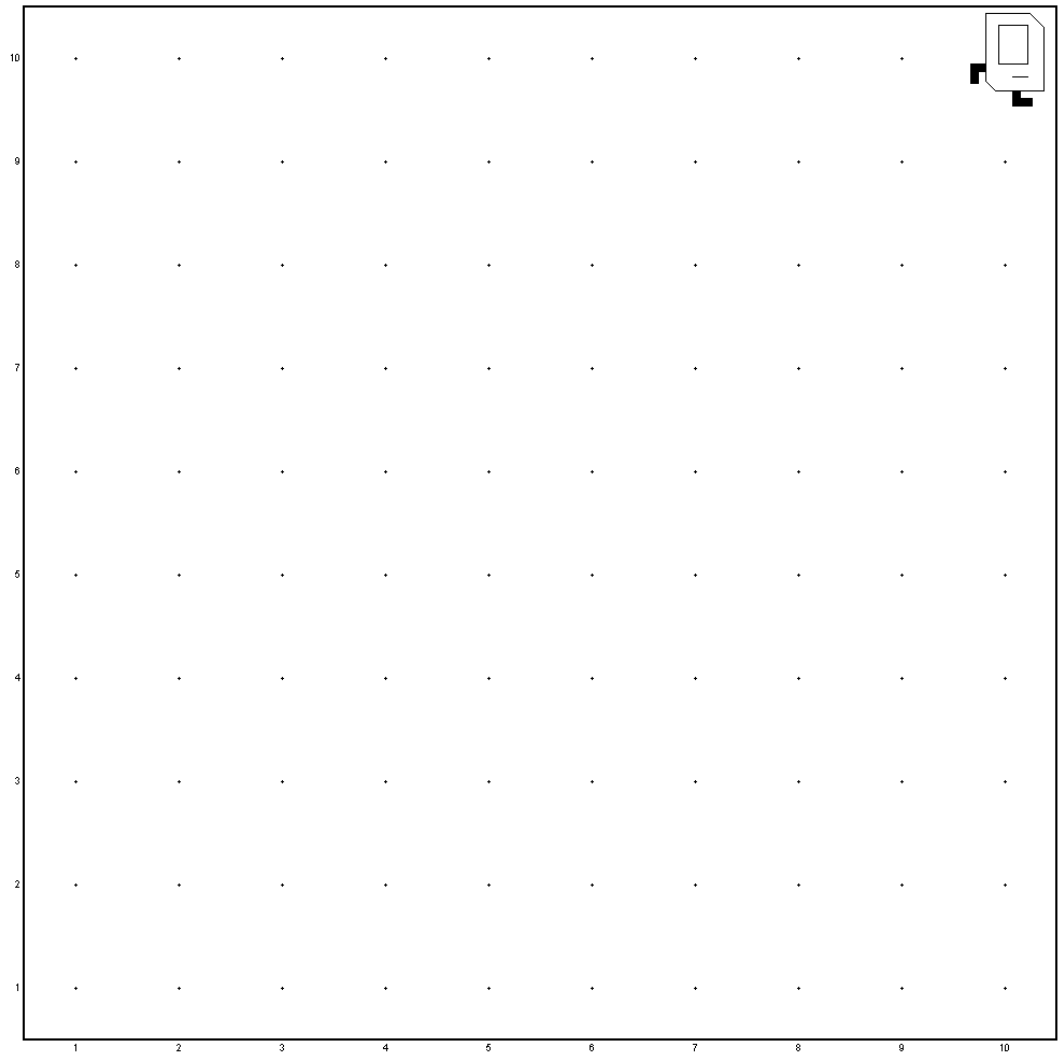


2) Diagonal algorithm

After the scanning is done, the Karel will check if the number of rows and columns are even and quale or not .

If the condition was true , then the Karle will use the diagonal algorithm to solve the problem .

The starting point for the algorithm is from the point (column counter, Row counter)



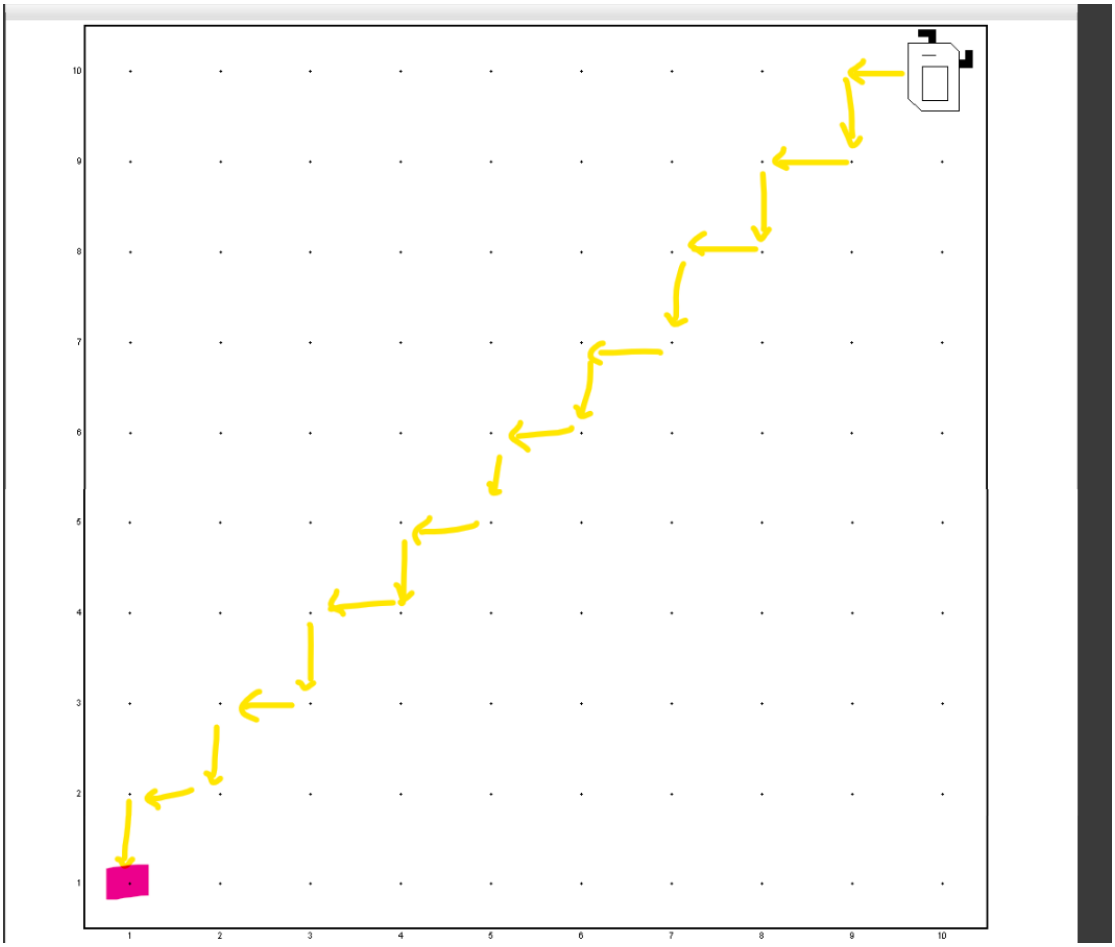
after determination of the number of both rows and columns, Karel will turn left to start the algorithm

my algorithm is to move the Karel diagonally since the rows and columns are equals .

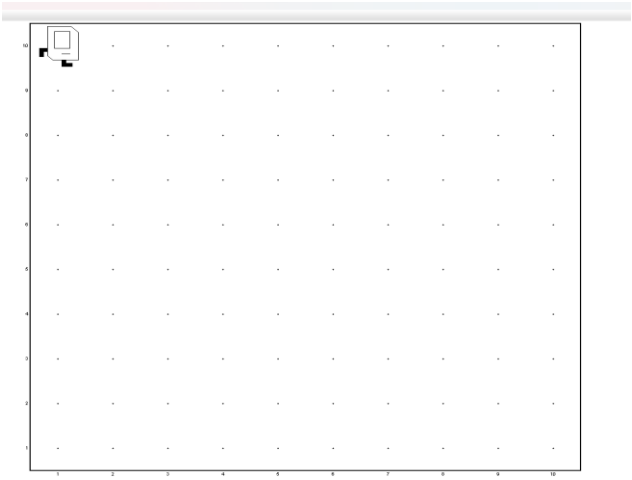
To make him move diagonally I defined a function to run the algorithm every time I need it

The steps to walk diagonally are :

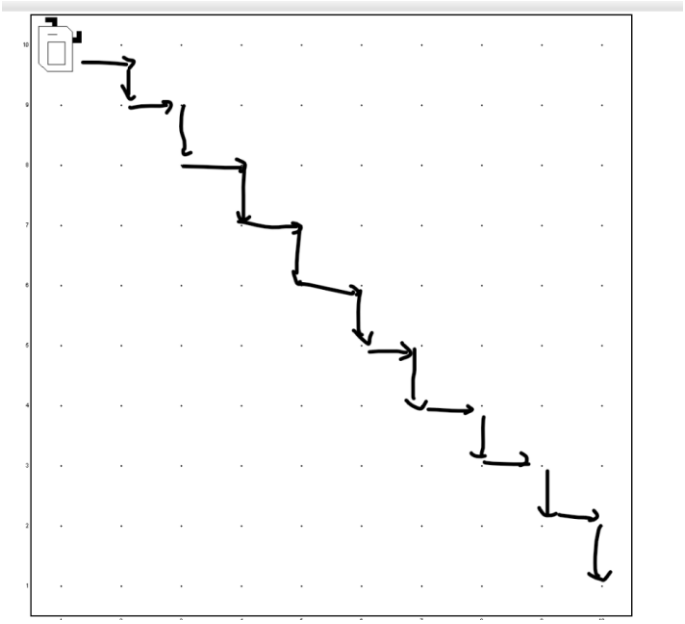
1- Make a loop to make it stop on the right point
2- Move a step
3- Turn left and move a step
4- Turn right
5- If no beepers are presented , put one



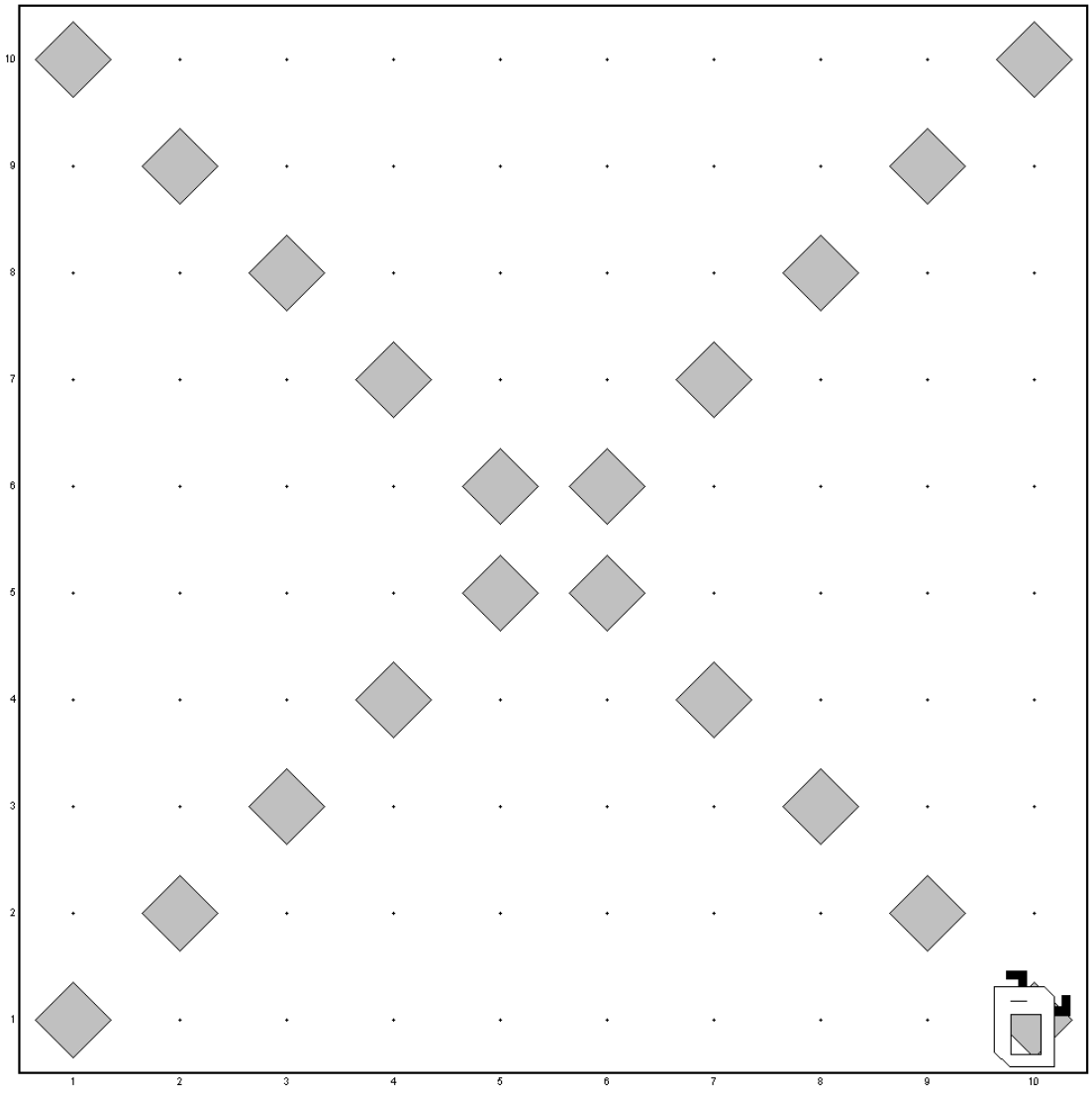
After the Karel draw the fist line of beepers, the Karel now on point (0,0) to make the other line of beepers on the other side , we need to move the Karel from 0,0 to (0,Row counter), then it will return right



Then we will reapply the previous steps to make the Karel move from the other side



After applying the diagonal algorithm



3)General Algorithm

After scanning Karel will check if the Rows are (even *Odd or versa), or odd*odd all both of Rows and columns are bigger than 2 then it will solve the problem using this algorithm.

As always the starting point after scanning is the most right corner of the map

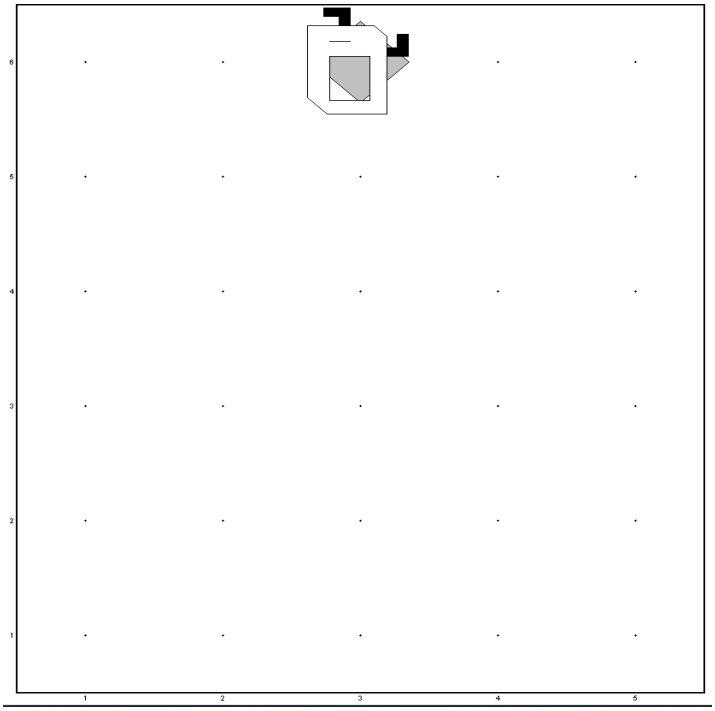
The general algorithm aim to divide the row depending on if the number is even or odd, if the number is even ,it will divide the number by 2 and add a line of beepers on result of his division and another line on the result +1

Otherwise if the number is odd it will add just one line of beepers all away (in horizontal way or vertical depending if its rows or columns)

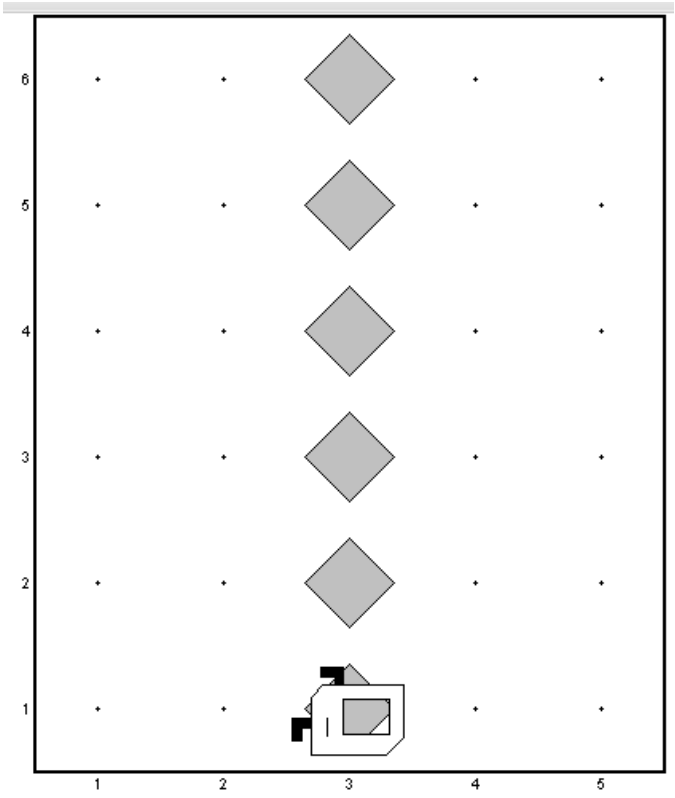
For example :

If the map contains 10 Rows and 5 column , what are the exact steps the Karel will do to solve the problem?

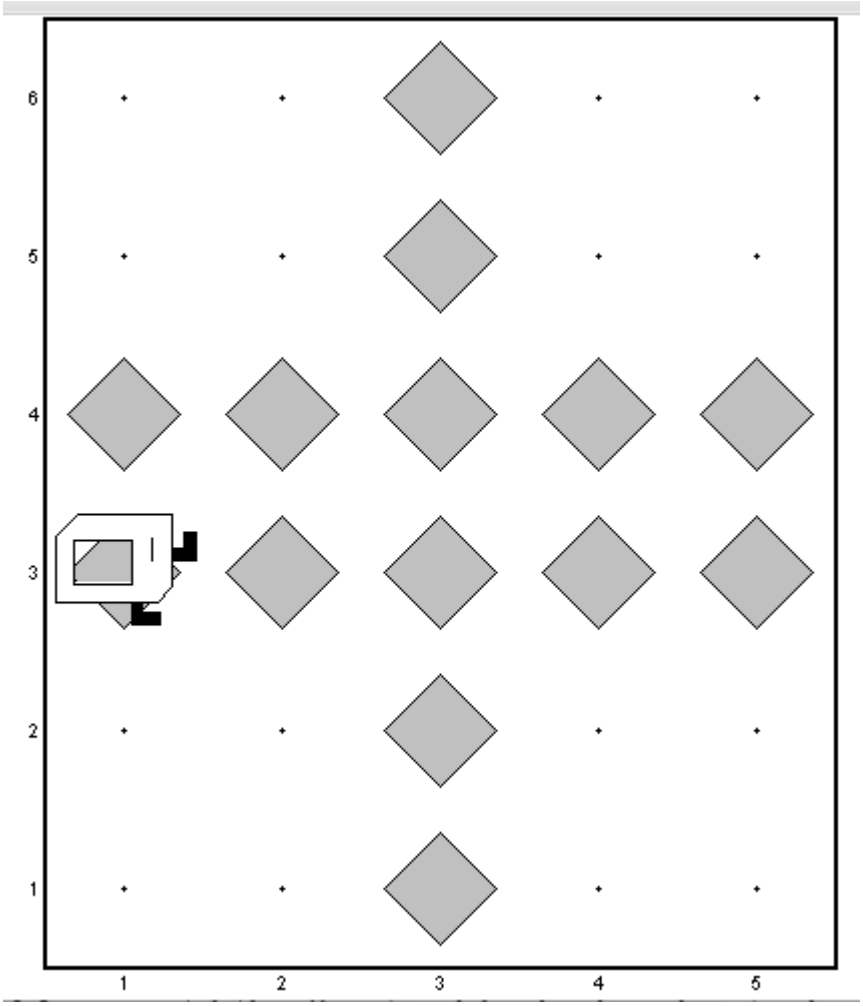
1) Karel will move to the middle of the column number, which is from the most right corner to the middle of the same row and put a beeper



2) Karle will move all through to the point (3,0), will putting a beeper with every move



3) Karel will turn around and move half the rows number
4) Turn left
5) Move until the front is blocked and put a beeper while moving
6) After it finishes the fist row, if the number of rows are even then Karel will turn left and add another line d beepers



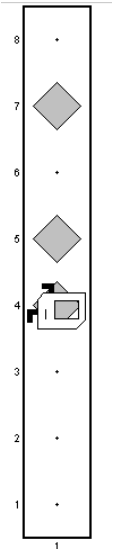
4) N*1 Algorithm

It solve the problem when we have a map with only one row and multiple columns or versa

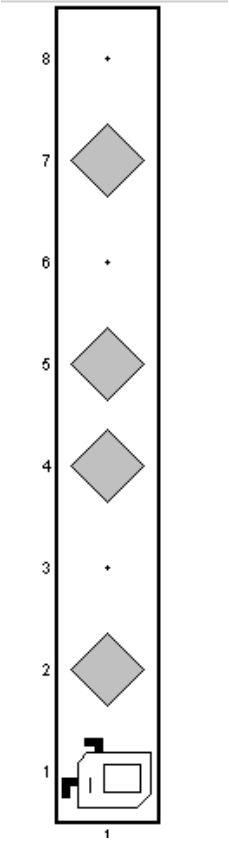
The algorithm depends on if the number is dividable by 4 or not , because it will make the Karel move ¼ of the number of rows or columns (the one which is bigger than one)

1- First Karel will check if the (Rows or columns) are dividable by 4 or not , if yes then
2- Move rows/4 steps and put a beeper

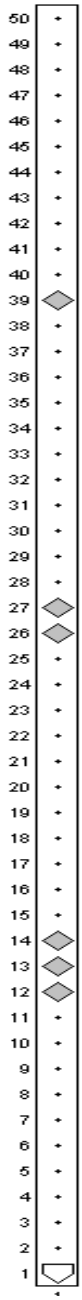
3- Check if Rows is even or not, is even move to Rows/2-1 and put a beeper and then move to Rows/2 and put a beeper



4- Move for another Rows/4 and put another beeper



if the number of rows are not dividable by 4 , the only deference is the loops and that here depending on the result of mod 4 I will put the beepers ,



Test cases

Number of Rows	number of columns	Number of steps	Number of Beepers	algorithms	Even or Odd	Equality of the number of columns and row
10	10	70	36	General	Even *Even	Equals
10	10	63	20	Diagonally	Even*Even	Equals
8	8	55	28	General	Even*Even	Equals
8	8	49	16	Diagonally	Even*Even	Equals
12	10	77	40	General	Even *Even	rows>coulmns
10	12	78	40	General	Even *Even	columns>rows
10	20	105	56	General	Even *Even	columns>rows
40	40	295	156	General	Even *Even	Equals
50	50	370	196	General	Even *Even	Equals
50	50	343	100	Diagonally	Evem*Even	Equals
4	3	18	8	General	Even*Odd	rows>columns
6	5	31	14	General	Even*Odd	rows>columns
10	3	33	14	General	Even *Odd	rows>columns
10	5	41	18	General	Even *Odd	rows>columns
10	23	113	54	General	Even *Odd	columns>rows
10	41	185	90	General	Even *Odd	columns>rows
50	49	317	146	General	Even*odd	rows >columns
3	10	37	14	General	Odd*Even	columns> rows
5	6	32	14	General	Odd*Even	columns>rows
7	24	84	36	General	Odd*Even	columns> rows
23	8	101	52	General	Odd *Even	rows>columns
49	50	318	146	General	Odd *Even	rows >columns
7	7	33	13	General	Odd*Odd	Equals
7	7	42	13	Diagonally	Odd*Odd	Equals
7	3	23	9	General	Odd*Odd	rows>columns
9	9	46	17	General	Odd*Odd	Equals
9	9	56	17	Diagonally	Odd*Odd	Equals
3	7	25	9	General	Odd*Odd	columns> rows
21	7	70	27	General	Odd*Odd	rows>columns
7	21	77	27	General	Odd*Odd	columns> rows
49	49	266	97	General	Odd*Odd	
49	49	336	97	Diagonally	Odd*Odd	Equals
8	1	13	4	1*N		even and divisible
49	1	86	5	1*N		Odd and not divisible
48	1	83	4	1*N		even and divisible
50	1	88	6	1*N		even and not divisible