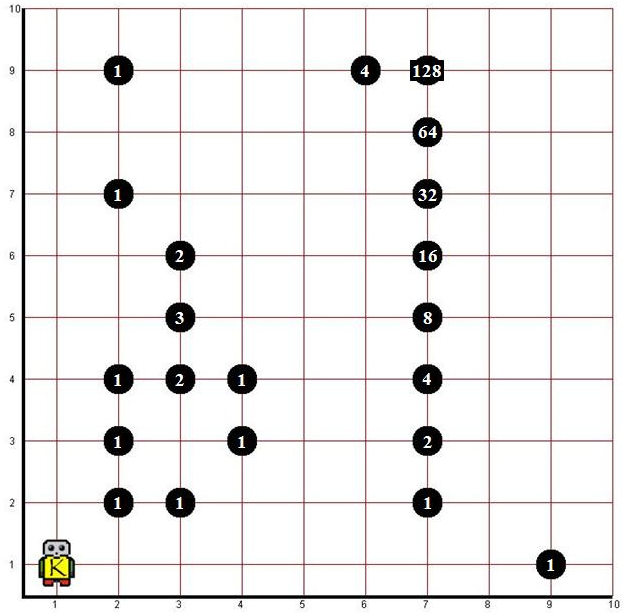
ColumnPicker

The world `randomColumns.kwld` contains a series of vertical columns. A column is never more than 8 streets high and starts at street 2. There are zero or more beepers at each intersection. Design a robot that can count the total number of beepers in a column, and display the result by having the robot place a pile of beepers (of that count) on street 1, directly below the column.



For example, on Street 1, Avenue 2, the robot should put a pile of 5 beepers. On Street 1, Avenue 3 the robot should put 8 beepers.

A beeper on street 1 indicates the end of the columns you must count. The state of the original columns must remain in place once the method completes.

How to Start:

Start by making a robot that can do the task for one column. Hint: start with the first column ☺

Here's a suggestion for a method you might have for counting one column:

```
public void countAndPlacePile()
{
      int total = countColumn(8);
      putNBeepers(total);
}
```

The above method assumes that you have a method called `putNBeepers(int numToPut)` that can put any number of beepers down. It also assumes that you have a method with the following signature:

```
public int countColumn(int numStreets)
```

which can presumably count the number of beepers in a column that is `numStreets` high and RETURN the count. The method must accept the number of streets because there are no other objects in the world that could be used to determine when to stop. This `countColumn` method is the crux of the whole thing.

THINK RECURSIVELY:

A beeper column that's 8 streets high, is just a beeper pile followed by a column that's 7 streets high….which is just a pile followed by a column that's 6 streets high…etc., etc. until you're asked to count a column that has no height. Stop and return the count. In other words, the beeper count for the column starting where the robot is standing is the count of what's beneath it, PLUS the count that's returned by counting the column starting one street further along.

EXAMPLE:
The count that should be returned for `countColumn(8)` would be the number of beepers the robot is currently standing on + `countColumn(7)` and so on. Note: the robot will have to move up the column as you go, just like `move(int numSteps)` the robot's movement must reflect the step of the recursion.

Encouragement:
This is a difficult recursive task that requires having your recursive methods return values, and having the recursive calls accumulate the result of those of those return values. There are a few ways to solve this problem both "on the way down" the recursive stack and "on the way back up." Think about the `findBeeper` problem we did in class which had the robot use "tail-end" recursion to get back to where it started. You can use tail-end recursion here to BOTH move the robot back down the column AND to accumulate the return values.