Simple exercises

1. Go to wall and place beeper

Karel should walk straight ahead till he reaches a wall. Upon doing so, he places a beeper. Test, e.g. on `empty_long.txt` (or `long_empty_start.km2` for the new interpreter).

2. Go to wall, place beeper, go back

The same as before, but after placing the beeper Karel should return to his initial cell and orientation. Test as before, but try different initial cells (or try `long_empty_middle.km2` for the new interpreter).

3. Move a pile of beepers

Assume that initially there is no wall in front of Karel and that the initial cell contains some beepers. The task for Karel is to move all these beepers to the cell in front of the original one. Test, e.g. on `tiny_city.txt` (or `tiny.km2`).

4. Move a pile of beepers at once

The same as the previous task, but during the whole program execution, Karel can perform the forward move only once (the forward move is the `STEP` instruction in the original implementation and the `move()` function in the new interpreter).

5. Copy a pile of beepers

Similar to the previous two exercises, but the task is now to copy the beepers instead of moving them. (I.e. if Karel starts in a cell with n beepers, in the end there have to be n beepers in the initial cell and n beepers in the "forward-neighbouring" cell.)

Arithmetic exercises

6. Summation

Karel is in the "`>0 n m`" situation (i.e. he stands in an empty cell and the next two cells in his line of sight have n and m beepers, respectively). The task is to get into the "`>(n+m) 0 0`" situation (i.e. Karel is back in his original position, but there are $n + m$ beepers in this cell and the next two cells are empty). Test, e.g. on `sum_city.txt` (or `1_sum.km2`).

7. Distribution

Karel is again in the "`>0 n m`" situation, but now the task is to get into the "`>0 (n+m)/2 (n+m)/2`" situation (if $n + m$ is even) or into the "`>1 (n+m)/2 (n+m)/2`" situation (if $n + m$ is odd, here / is the integer division). Test on the same benchmarks as before or on `2_dist.km2`.

8. Integer division with a remainder

Karel is in the `>a b` situation. Assume that all the cells around these two cells are empty and without a wall. The task is to get into the following situation:

```
a b
r q
```

where $q = a \mathit{div}\, b$ and $r = a \mathit{mod}\, b$. All the places not pictured in the target situation have to be empty (but you can use them to store beepers during the computation).You can assume $b \neq 0$. Test, e.g. on `division_city.txt` (or `3_div.km2`).

Advanced exercises

9. Staircase

Karel is in a long empty corridor and his task is to build a "staircase", i.e. get from the ">`0 0 0 ... 0`" situation into the ">`1 2 3 ... n`" situation. You can assume that Karel initially stands with his back towards a wall (though this assumption is not necessary for writing a correct program). Test, e.g. on `corridor.txt`, `empty_long.txt` (or `long_empty_start.km2` and `long_empty_middle.km2` for the new interpreter).

10. Maze exploration

Karel's map does not contain any beepers. The task is to explore the whole map and put a **single** beeper into every cell reachable from the original one. For the old version, test on `cityNN.txt` for the available choices of `NN`. For the new version, test on `NN.km2` from the "empty" folder. Try various initial cells.

11. Treasure hunt

The same situation as before, but the map contains a treasure: a cell initially marked by two beepers. The task is to reach the treasure and stop in the cell containing it. The other cells can, at the end, contain an arbitrary number of beepers, though we are guaranteed that they are initially empty. For the old implementation, test on `cityNN.txt`, (you have to write 2 into one of the cells). For the new interpreter, you can use `NN.km2` from the "treasure" folder.

12. Lightweight tidying

Karel's map is full of trash! Some places may initially contain a beeper (at most one). The task is to tidy up, i.e. to ensure that in the end, the map contains no beepers. Test on `cityNN_dirty.txt` (for the new version, there are unfortunately no pre-made benchmarks available, so just modify the previous benchmarks by randomly throwing in some beepers).

Open problem

Heavy tidying

What if the cells can initially contain an arbitrary amount of trashy beepers? How would you handle the tidying then?