Most of this week's section is about meeting your section leader, meeting other students in your section, and then getting some practice with Karel problems before you are faced with them on Problem Set 1. As such, there are two problems included here: one "what does Karel do" code comprehension type problem, and then a problem to practice writing Karel code to solve a task.

1. Consider the following Karel program:

```
import karel

def mystery():
    for i in range(4):
        while front_is_clear():
            put_beeper()
            move()
        turn_left()
```

   and then answer the following questions.

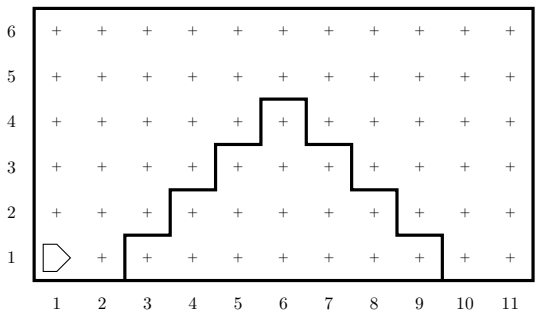
   (a) What does this program accomplish if Karel starts facing east in the lower left corner of an otherwise empty world? You can assume Karel starts with an infinite amount of beepers in their bag.

   (b) What might be a better name for this function?

   (c) A good name can help with readability, but you should always include a docstring as well describing what a function does. What might that docstring comment say here?

   (d) Whenever possible, it is often a good idea to reuse code that you or others have already written. Is there any code or useful functions that you have already seen in class that might be able to accomplish the same thing as parts of this code?

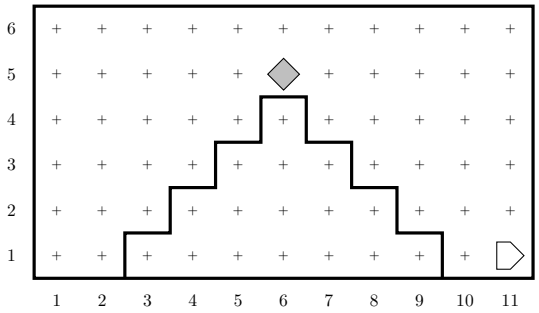
2. Here you task is to write a program that teaches Karel to climb a "stair-step" mountain. An example of such a mountain is shown below, but your program should be general enough so that Karel could climb a mountain of any size.



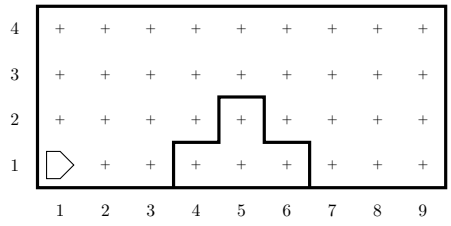
   The steps involved in such an endeavour might look like:

1. Move up to the start of the mountain.
2. Climb each of the stair steps to reach the summit.
3. Plant a flag (represented by a beeper, naturally) at the summit.
4. Climb down each of the stair steps on the opposite side.
5. Move forward to the east end of the world.

The final state of the world should thus look like:



If you knew in advance that there were exactly four steps, then you could use a `for` statement to repeat the statements required for each step. In the general case, however, you have to figure out how Karel can determine when it has reached the summit, and then later how it has reached the ground again. Your program should be able, for example, to scale a molehill like:



or an Everest-sized peak like: