ADVANCED PYTHON HOMEWORK 4

Relax with a puzzle game

Choose one of the following tasks and solve it. Make sure your solution does not output or generate needless lists (such as a list of all solutions). Every task is worth 4 points.

Lab teacher's notes: Do not forget to test your solutions with a reasonable amounts of tests, and include them in the homework. Time measurements and determining how large inputs are still feasible for your solution is also strongly recommended. If you ever make assumptions that are not specified in the exercise, make sure that they are well documented.

Exercise 1. Cryptarithm is a puzzle where letters represent (unknown) digits and you need to find a mapping such that the described mathematical operations are completed successfully. An example of a (rather simple) cryptarithm is below:

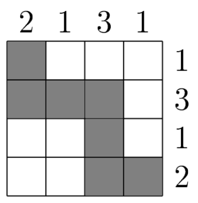
```
  KIOTO
+ OSAKA
-------
  TOKIO
```

Other examples can be found e.g. at `https://www.math.edu.pl/kryptarytmy`.

Write a function that finds a solution to a cryptarithm, if it exists.

Exercise 2. We wish to reconstruct a picture based on "casting a shadow". This is also called a "nonogram" or "picross" in English. We start with a blank two-dimensional grid, where a top row and a top-right column have an additional vector of shadows. The goal is to find a black-and-white picture such that each row and each column contains the shadow (an interval of black squares) of exactly the requested size.

Write a function that finds a solution to a nonogram, if it exists. Here is a (small) example for two vectors $H = (2, 1, 3, 1)$ and $V = (1, 3, 1, 2)$:



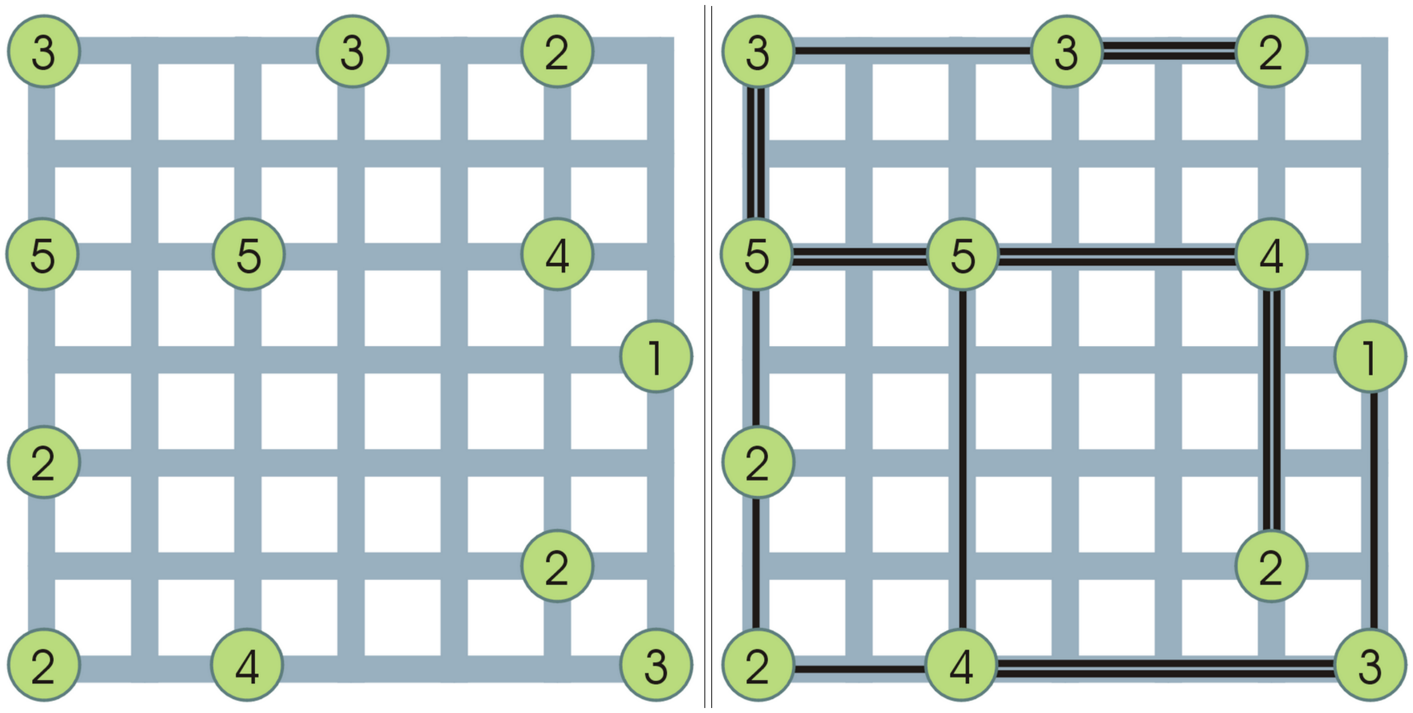
You can find more about nonograms at `https://en.wikipedia.org/wiki/Nonogram`.

Exercise 3. Everyone knows the Sudoku puzzle – we are given a 9×9 grid and you are tasked to fill in digits from 1 to 9 so that every row, every column and each of the 9 major 3×3 sub-squares contains exactly all digits from 1 to 9. Below is an example of a fully solved Sudoku:

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Write a function `solving_sudoku(s)` that tries to finalize a partially filled diagram s or returns `None` if there is no solution. (Note: the input should ideally be a list of lists, corresponding to a list of all rows, with None where no number is pre-set.) Alongside this, implement another function that nicely draws a partially filled or a fully filled input diagram.

Exercise 4. Build some bridges! (橋をかけろ, hashiwokakero) This is another puzzle based on a diagram like this:



The rules of the bridges, borrowed from English Wikipedia, are as follows:

- They must begin and end at distinct islands, travelling a straight line in between.
- They must not cross any other bridges or islands.
- They may only run orthogonally (i.e. they may not run diagonally).
- At most two bridges connect a pair of islands.
- The number of bridges connected to each island must match the number on that island.
- The bridges must connect the islands into a single connected group.

Your task is yet again to implement a function `hashi(s)`, which is given a list-of-lists representation of the puzzle, and needs to produce at least one solution for the problem, if it exists. You can find more about this puzzle online, e.g. on Wikipedia: `https://en.wikipedia.org/wiki/Hashiwokakero`.