Logic and Computability SS22, Practical Bonus Assignment 2

Due: TBD

[5 Points] *Magic Square.* For this programming exercise we will have a look at the classic pen and paper puzzle called *magic square.* The premise of the game is very intuitive. You are given a grid of numbers, where some of them might be missing. Your goal is to determine the missing numbers so that the sum in each *row*, *column*, and *diagonal* is the same. In a sense, you are trying to *balance* out the square of numbers. If you want to get a better feeling for the problem, you can play around with the Magic Square app from the Play Store.
The magic square is read from a file, where the numbers are separated by spaces and unknown cells are marked with ”_”. For the `test0.txt` the resulting magic square is shown in Figure 1.
Note that the size of the grid is not fixed but may vary, whilst always going to be square.

		28
	35	21
	7	

Figure 1: Example of a magic square.

Our goal is then to find the missing cells and fill out the square. This is where the SMT solver can help us! An example solution is shown in Figure 2. Here is a short list of tasks you need to implement in `square.py` when solving this programming exercise.

14	63	28
49	35	21
42	7	56

Figure 2: Solution of the example magic square.

1. Create a Z3 integer variable for each of the cells in the magic square. You should probably give them meaningful names like `C_0_0`, `C_1_2` or similar. This will make it easier to debug.
2. Enforce that the known numbers have the expected value. This one is self explanatory, since we cannot change the value of the predefined cells.
3. Create a *magic* Z3 integer variable which will hold the sum. This variable is used to enforce that all the rows, columns and both diagonals add up to the same number.
4. Enforce that all columns add up to the *magic* variable.
5. Enforce that all rows add up to the *magic* variable.
6. Enforce that both diagonals add up to the *magic* variable.

Everything else is already implemented in the template that you can pull from our upstream repository. For details on how to do that, please consult the provided `README.md`. The template handles the parsing of the input file, and the printing of the solved square. The only additional Python library is `z3-solver` which you need to install through `pip install z3-solver`. If you get stuck, you can always drop us a message in our Discord channel. Keep in mind that the implementations of the subtasks are often no more than two or three lines of code, so avoid thinking of super non-obvious solutions. *Good luck and have fun with the exercise.*