{C} TRY CATCH

Chess Challenge

Below you will find the description of the problem. Using the agreed upon language (no special libraries are needed but you can use external ones if you feel such need), please provide an answer for the question at the end. You should be able to finish the assessment within one week. If you cannot find the solution, please do not worry; it's not an easy task and we would like to see your thinking and problem solving abilities more than an accurate answer.

Please commit your code changes to Git repository (Github, Bitbucket, it's up to you. We would like to view the code afterwards in a browser and clone it to run it). By seeing your commits we grasp the way you work and by reviewing your code we learn more about your technical skills.

Afterwards you will receive solid feedback about your code and if everything is fine, you will be invited to the next stage. **Please note that in case we are all good and ready to propose you to the best IT companies we work with, the code you will ship for this assessment will be presented to them too.** That's why it's really important you will put your heart into it. Should you have any questions or concerns regarding the assessment, please do get in touch.

Problem

The problem is to find all unique configurations of a set of normal chess pieces on a chess board with dimensions M×N where none of the pieces is in a position to take any of the others. Assume the colour of the piece does not matter, and that there are no pawns among the pieces.

Write a program which takes as input:

- The dimensions of the board: M, N
- The number of pieces of each type (King, Queen, Bishop, Rook and Knight) to try and place on the board.

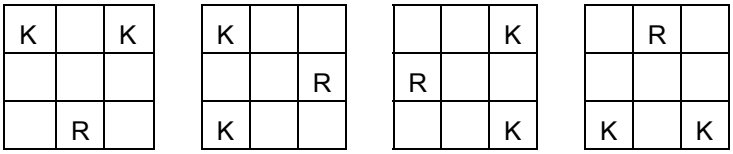
As output, the program should list all the unique configurations to the console for which all of the pieces can be placed on the board without threatening each other.

When returning your solution, please provide with your answer the total number of unique configurations for a **7×7 board with 2 Kings, 2 Queens, 2 Bishops and 1 Knight. Also provide the time it took to get the final score. Needless to say, the lower the time, the better.**

Examples

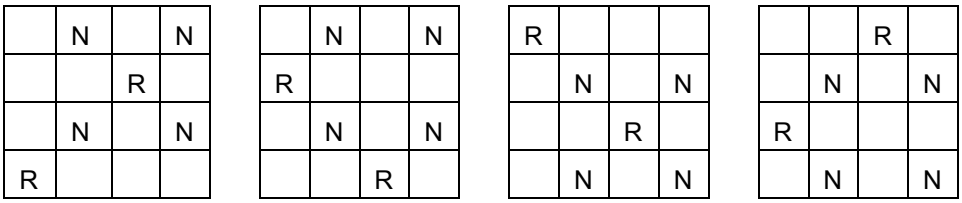
Input: 3×3 board containing 2 Kings and 1 Rook.

Output:



Input: 4×4 board containing 2 Rooks and 4 Knights.

Output:



Hints

Please follow best practices whilst writing your code. Tests, proper commits, proper configuration, good code structure, clean code practices are among others, good signs of your professional approach. Your code will be used as a proof of your capabilities. Below you can find some important hints (feel free to comment on them and send your remarks in case you have any)

- Code has to comply to https://www.python.org/dev/peps/pep-0008/
- Code should be covered with unit tests in at least 90%
- Pylint score should be at least 9.0/10
- docstrings should comply to pep257 for every public class and method and module function (except from tests)