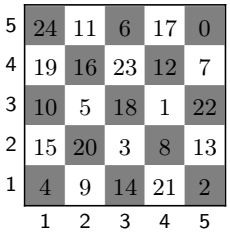
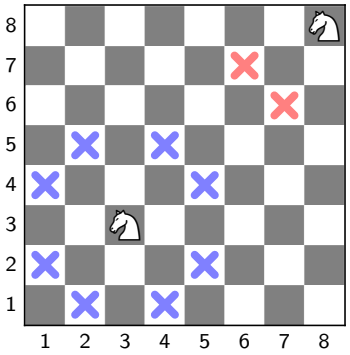
Coursework 2 (Knight's Tour)

This coursework is worth XXX% and is due on 21 November at 16:00. You are asked to implement a Scala program that solves the knight's tour problem on an $n \times n$ chessboard. This problem is about finding a tour such that the knight visits every field on the chessboard once. One a 5×5 chessboard, a knight's tour is as follows:



The tour starts in the left-upper corner, then moves to field $(4,3)$, then $(5,1)$ and so on. A knight's tour is called closed, if the last step in the tour is within a knight's move to the beginning of the tour. So the above knight's tour is not closed (that is it is open) because the last step on field $(1,5)$ is not within the reach of the first step on $(5,5)$. It turns out there is no closed knight's tour on a 5×5 board. But there is one on a 6×6 board.

If you cannot remember how a knight moved in chess, below are all potential moves indicated for two knights, one on field $(3,3)$ and another on $(8,8)$:



Disclaimer

It should be understood that the work you submit represents your own effort. You have not copied from anyone else. An exception is the Scala code I showed during the lectures or uploaded to KEATS, which you can freely use.

Task

The task is to implement a regular expression matcher based on derivatives of regular expressions. The implementation should be able to deal with the usual (basic) regular expressions

Important! Your implementation should have explicit cases for the basic regular expressions, but also explicit cases for the extended regular expressions. That means do not treat the extended regular expressions by just translating them into the basic ones. See also Question 2, where you are asked to explicitly give the rules for nullable and der for the extended regular expressions.