Exercise 1 (At most one with \( \log(n) \) extra variables) [4 points]

Show how to encode the at-most-one constraint over \( n \) Boolean variables by only using \( O(\log n) \) extra variables and \( O(n \log n) \) clauses.

Exercise 2 (Pigeon Hole Principle) [4 points]

Show how to encode the following question into a SAT formula: is it possible to put \( n + 1 \) pidgeons into \( n \) holes such that each pidgeon has its own hole?

Exercise 3 (Planning Encoding) [7 points]

Design and describe a SAT encoding for SAS+ planning that uses only one kind of Boolean variables – variables representing actions (\( a_i = \text{True} \) if and only if \( a_i \) is present in the \( t \)-th step of the plan). Compute how many clauses are required for such an encoding.

Exercise 4 (Sliding Puzzle Challenge) [10(+10) points]

Implement a DIMSPEC encoder for the sliding puzzle problem. Your application should take a single command line argument – a string with the initial state of the puzzle (example: "123-480-765", goal state is always "123-456-780", "0" represents the empty slot) and print the sequence of steps required to solve it (movements of the empty slot, in the example above: "DLURD"). The puzzle will have at most 15 pieces (0123456789ABCDEF) and is not necessarily a square.

Some example inputs for testing: "023-145", "1034-9267-D5B8-E6FC", "162-530-478", "012-345-678", "876-543-210", you can create your own inputs, be careful not to create unsatisfiable ones :) Use Stephan’s DIMSPEC solver (called incplan in the IPASIR app package: [https://github.com/biotomas/ipasir](https://github.com/biotomas/ipasir)) to solve DIMSPEC files. The best solver (solving the most instances and fastest) will get a bonus of 10 points.

Exercise 5 (Sokoban Challenge) [15(+15) points]


- benchmark levels: [http://baldur.iti.kit.edu/sat/files/sokoban.zip](http://baldur.iti.kit.edu/sat/files/sokoban.zip)
- DIMSPEC solver: incplan app in [https://github.com/biotomas/ipasir](https://github.com/biotomas/ipasir)