Correction of proposed exercises

1. Three thieves have robbed a wine barrel of 24 liters. They would like to divide the wine in three equal parts (8 liters each). Unfortunately, they only have at their disposal three vessels: one of 5 liters, one of 11 liters and one of 13 liters.

Write a prolog program to solve this decanting problem.

2. 8-puzzle (sliding puzzle):
The 8-puzzle is a smaller version of the slightly better known 15-puzzle.
The puzzle consists of an area divided into a grid, 3 by 3 for the 8-puzzle (4 by 4 for the 15-puzzle). On each grid square is a tile, except for one square which remains empty. Thus, there are eight tiles in the 8-puzzle. A tile that is next to the empty grid square can be moved into the empty space, leaving its previous position empty in turn. Tiles are numbered, 1 to 8 for the 8-puzzle, so that each tile can be uniquely identified.

The aim of the puzzle is to get the configuration where all the tiles are ordered from any given starting configuration.

Write a prolog program to solve this puzzle.
(Hint: the puzzle doesn’t always have a solution !)

3. You are a train driver. Your train is represented by a list of the form \([c_1, c_2, ..., c_n]\) where \(c_i\) are the cars. The locomotive is supposed to be on the left of the car \(c_1\) but is not explicitly represented.

You are in a marshalling yard and your task is to rearrange the cars in a specific order. The marshalling yard has two sorting tracks where you can push or remove cars.
Here is an example of the rearrangement of the train \([a, b, c]\) to \([b, c, a]\).

Write a prolog program to compute the movements necessary to rearrange a train.

**Exercises**

4. Define a predicate `listOfPred(+Pred(...), -Ls)` that succeeds if the list `Ls` is a list where the first element is the predicate `Pred` and the next are the arguments of the predicate `Pred`.

```prolog
?- listOfPred(between(1, 100, X), Ls).
   Ls = [between, 1, 100, X].
```

5. Define a predicate `applyPred(+Pred, +Args)` that succeeds if `Pred(Args)` is true.

6. Define a predicate `filter(+L1, +Pred, -L2)` that succeeds if `L2` is the list of `L1`'s elements that satisfy the predicate `Pred`.

**Memoization**

7. Define a predicate to compute the function \(f\):

\[
f(n) = n \text{ si } 0 \leq n \leq 2
\]

\[
f(n) = [2f(n - 1) + 3f(n - 2) + 5f(n - 3)] \mod (n + 1) \text{ si } n > 2
\]