

assignment #4 (winter term 2005)
solutions will be presented Tuesday, 22-Nov-2005, 2 PM, o27/2203
<http://www.informatik.uni-ulm.de/pm/index.php?id=112>

Exercise 1 (Warmup – generate and test).

- a) Implement a Prolog-predicate `permutation/2` that generates permutations: `permutation(A,B)` generates a permutation B of a list A with fixed length. All permutations can be computed by backtracking using “;” when prompted. Do not use built-in predicates for manipulating lists. Describe your observations with the following queries:
- `permutation([1,2,3],[L,M,N])`.
 - `permutation([L,M,N],[1,2,3])`.
 - `permutation([1,2,3],B)`.
 - `permutation(A,[1,2,3])`.
- b) Improve your program, such that *all four* queries from (a) work.

Exercise 2 (*n*-Queens Example – generate and test).

Place *n* queens on an $n \times n$ chessboard, s.t. no queen attacks any other queen.

As each row must contain exactly one queen, a representation can be given by a list (length *n*) of X-coordinates. Generate all possible positions and check. If the placement is not save, backtrack.

Write a Prolog-program `solve(n,L)` which solves the *n*-queens problem for arbitrary *n* and returns a list L of X-coordinates of the *n* queens.

Hint: The solution given in the introduction uses CHR code – here we try it in Prolog.

Exercise 3 (CLP – “generate and test” vs. “constrain and generate”).

Use the *permutation sort* algorithm for sorting a list of integers. Analyse the run time complexity of each implementation wrt. to the length of the list.

- a) To implement the *generate and test version*, use three Prolog predicates `permsort(List,Sorted)`, `permutation(List,Sorted)`, and `sorted(Sorted)` (all arguments are lists). A list R is the sorted version of the list L if R is a permutation of R and the elements or R are sorted in increasing order.
- b) For the *constrain and generate version*, use the constraint solver given in the modul `clpq` and refer to the SICStus manual – chapter 33 – for details.

Hint: Your source code must include `:- use_module(library(clpq))`.

For example, the constraint solver simplifies the following two constraints (which must be enclosed in braces) `{A=<3, A=<5}` to `{A=<3}`.

Convert, both the explicit as well as the implicit compare operations (i.e., the operations that compare list elements) from part (a), into `clpq` constraints by attaching braces.

Note that in the definition of `permsort`, the predicate `sorted` should be called before `permutation` in order to constrain first.

Testing should include lists with variables (e.g. `permsort([1,A,3],Sorted)`) and already sorted (output)lists (e.g. `permsort(X,[1,3,7])`).

If you run into an endless loop, explain why and fix the problem.