Some Solutions to the N Queens Problem

Author: Libor Spacek (C) <libors@gmail.com>

Date: 21th May 2008 Version 2: May 2009

Example usage:

```
# pure -i queens.pure

>fullboard (thequeens 50);
```

(fullboard s)

full coordinates representation [(row,column),...] is redundant because the first coordinate of any solution will always be 1..n. However, it can be reconstructed with, e.g.: (fullboard (thequeens n)); By default all solutions here are encoded as the columns permutations only, [column, ...], leaving out the ordered rows. In all cases rows and columns are interchangeable without affecting the solutions, i.e. reading the same numbers as [row, ...] for ordered columns is also valid.

(allqueens n)

returns all possible solutions by constrained search. N.B. nobody has so far been able to find out, by any known method, the number of solutions for n=26 and beyond. Allqueens, which is quite fast, starts getting breathless beyond n=13 with #solutions = 73,712. Only half of the first row is considered for the first queen, thus halving the total search effort, followed by adding a reflection of all the solutions found about the middle column. Odd sized boards must have additionally the first midrow starting position searched and add its solutions. All rook checking (row and column) is eliminated by using only unused candidates list c (generating only valid permutations), leaving just a single check, id==abs(j2-j), to perform for the bishop (both diagonals). More reflections/rotations could be done but they would have to be tested for duplicates.

```
>allqueens 8; returns all 92 solutions, as a list of lists
```

(queens n)

(tailqueens n)

this concise backtracking tailrecursive version throws a single solution which is the rows reflection of that found by "queens"

(thequeens n)

encodes my no search regular solution in just 12 lines of code, which is to my knowledge the simplest and fastest known algorithm for the N-Queens problem. It is very fast even for large boards.

There always exists one symmetrical (under 180 degrees rotation) solution of this form, producing an orbit of just 4 equivalent solutions, instead of the usual 8. The correct pattern is generated directly without any checking or searching being necessary. The solutions had been tested exhaustively for board sizes 0 to 5000 and individually for board size 50000x50000.

(checkqs l)

checks one solution either in 0..n-1 encoding or in 1..n encoding. It returns 1 for a correct result, including "nosolution" for sizes 2 and 3; 0 is returned if a queen attack exists anywhere within the presented 'solution'.

(queenstest method l)

conducts exhaustive tests of solutions for boards of all listed sizes. Usage:

```
>queenstest (id) (allqueens 8);
>queenstest queens (1..10);
>queenstest tailqueens ([5,6,7]);
>queenstest thequeens (5000:4999..100);
```