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);
```

(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 the first row is used for the first queen, thus halving the total search effort, followed by a single reflection of the solutions found. Odd sized boards have in addition the middle row searched separately. All row and column checking is eliminated by passing valid candidates c (generating only valid permutations), leaving just a single check to perform for 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)

the single solution is encoded as the columns permutation, leaving out the ordered rows i: (1..n). Full 2D board coordinates can always be reconstructed with zip (1..n) (queens n);

```
>queens 8; gives solution number 52 in the allqueens' list.
```

(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, 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, consequently producing an orbit of just 4 equivalent solutions, instead of the usual 8. These few lines of code are self-contained (not calling any square checking). The solutions had been tested exhaustively for board sizes 0 to 5000 and also individually for board size 50000x50000.

Row numbering in 'thequeens' is changed for simplicity to 'C style' 0..n-1. Solution using 2D board coordinates (1..n)x(1..n) can be easily reconstructed with: (**fullboard** (thequeens n)). Note - *nosolution* is correctly returned for n=2 and n=3

```
>map succ (thequeens 8); gives solution no. 56, encoded 1..n
```

The rest are test utilities:

(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 an exhaustive test 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);
```