Exercise 2.43.

Louis Reasoner is having a terrible time doing exercise 2.42. His `queens` procedure seems to work, but it runs extremely slowly. (Louis never does manage to wait long enough for it to solve even the 6×6 case.) When Louis asks Eva Lu Ator for help, she points out that he has interchanged the order of the nested mappings in the `flatmap`, writing it as

```
(flatmap
 (lambda (new-row)
   (map (lambda (rest-of-queens)
          (adjoin-position new-row k rest-of-queens))
        (queen-cols (- k 1))))
 (enumerate-interval 1 board-size))
```

Explain why this interchange makes the program run slowly. Estimate how long it will take Louis's program to solve the eight-queens puzzle, assuming that the program in exercise 2.42 solves the puzzle in time T.

Answer.

To explain why Louis's procedure drops behind so giantly, let's first analyze the performace of the original one

```
(flatmap
 (lambda (rest-of-queens)
   (map (lambda (new-row)
          (adjoin-position new-row k rest-of-queens))
        (enumerate-interval 1 board-size)))
 (queen-cols (- k 1)))
```

Suppose the problem we deal with has a size of N, that is, the value of `board-size` here is N. `Queen-cols` invokes itself once with one less size of problem when evaluated until it reaches the base case where $n = 0$. Thus, in evaluating the expression `(queen-cols n)`, a tree of depth N is built, and each level sits only one node. So the order of growth of the number of steps used by this process is

$$T = \Theta(n)$$

By swapping the order of the nested mappings in the `flatmap`, Louis's procedure calls itself N times with each decrement of the size of the problem. So what built by his procedure is an complete N-nary tree of depth N. Hence, it will takes up approximately

$$T = \Theta(n^n)$$

steps to evaluate `(queen-cols n)` using Louis's procedure. This giant order of growth dramatically but strongly illustrates why Louis's implementation has such a bad performance.

*. Creative Commons 2013, Lawrence R. Amlord(颜世敏).