# Programming Paradigms

Lab 4. More exercises in Racket

## Outline

- Higher-order functions and lists recap
- Exercise: Eight queens puzzle

## Eight Queens puzzle: preliminaries

**Exercise 4.1.**
Implement function **`attacks?`** that checks whether two queens attack each other. Represent a queen with a pair or a list of coordinates.

**Exercise 4.2.**
Implement function **`attacks-any?`** that checks whether a queen attacks any of the other queens (given as a list).

**Exercise 4.3.**
Implement function **`no-attacks?`** to check whether a given arrangement of queens has no two queens attacking each other.

## Eight Queens puzzle: naive solution

**Exercise 4.4.**
Implement function **`for-range`** that applies a given function to every number in a given range and returns a list of results.

**Exercise 4.5.**
Implement function **`naive-four-queens`** that finds all solutions to Four Queens puzzle (on a 4x4 board) by iterating over all possible row positions for four queens located on different columns and checking the corresponding solution.

**Exercise 4.6.**
Suggest optimizations for the implementation in Exercise 4.5.

## Eight Queens puzzle: finding all solutions

**Exercise 4.7.**
Implement function **`add-queen-at`** that takes a list of possible N queen arrangements on a 8x8 board and returns a list of possible N+1 queen arrangements, where the last queen is added at a given column.

**Exercise 4.8.**
Implement **`eight-queens`** that finds a solution for Eight Queens puzzle.

**Exercise 4.9.**
Generalize implementations from Exercises 4.7 and 4.8 and implement function **`n-queens`** to find all solutions to N Queens puzzle.

## Eight Queens extra: generating solutions one by one

**Exercise 4.10.**
Use **`generator`** and **`yield`** (see Racket Guide 4.15.3) and implement **`for-range-gen`** that applies a function to every number in a given range and yields results one by one.

**Exercise 4.11.**
Use **`for-range-gen`** to implement **`eight-queens-gen`** that yields solutions to the Eight Queens problem one by one.