Topics Today

- n-queens problem (Continuation-based Solution)
- Exceptions
- n-queens problem (Exception-based Solution)

$$\Downarrow$$

Continuations / Exceptions

Continuations - Recap

- functional argument representing the "effect of the rest of the program"
- initial continuation: fn x =>x , print
- **Theorem**: For all expressions e and all continuations k,

$$\mathsf{eval}'(e)\ k = k(\mathsf{eval}(e))$$

$$\Downarrow$$

tail-recursive, more efficient

n-Queens Problem - 1

n-Queens Problem:

place n queens on a $n * n$ board in such a way that no two queens are attacking each other

Threat:

- along rows
- along columns
- along diagonals

n-Queens Problem - 2

Task: Write a function

queens: int$\rightarrow$ placement list

which returns all possible placements

Data Types:

- type position $= \underbrace{\mathsf{int}}_{\text{row}} * \underbrace{\mathsf{int}}_{\text{column}}$
- type placement = position list

Algorithm for n-queens problem

Exceptions

- Catch potential run time errors explicitly
  (for example: divide-by-zero)
- Define exceptions:
  ```
  exception DivZero
  exception NoEntry of string
  ```
- Raise exceptions:
  ```
  raise DivZero
  raise NoEntry ("Bovik, Harry")
  ```
- Handle exceptions:
  ```
  < exp > handle DivZero => < exp >
  < exp > handle NoEntry (x) => < some exp using x >
  ```


Using Exceptions

```
exception DivZero

val Infinity = 65,556

fun divide (x,y) =
   if y = 0 then
     raise DivZero
   else
      x div y

fun div1 (x,y) = divide (x,y) handle DivZero => Infinity
```


n-queens using exceptions

Task: Write a function

$\boxed{\text{queens3: int} \rightarrow \text{placement}}$

which returns *one* possible placement, if there is one

for all rows, try to find a safe position in row i
if there is no safe position in row i, then backtrack

Another solution to the n-queens problem

```
fun board i [ ] = [ ]
  | board i (j::js) = (i,j)::(board (i + 1) js);
fun ok [ ] = true
  | ok (q::qs) = (safe q qs) andalso ok qs;
fun filter p [ ] = [ ]
  | filter p (x::l) = if p x then x::(filter p l) else (filter p l);
fun upto i n = if i > n then [ ] else i::(upto (i+1) n)


fun queens2 n = filter ok (map (board 1) (perm (upto 1 n)))
```


Algorithm for n-queens using exceptions

```
fun queens3 n =
    let exception Conflict;
      fun addqueen i qs =
        let fun try j =
          (if not(safe (i,j) qs) then
              raise Conflict
           else
              if i = n then
                 (i,j)::qs
              else
                 addqueen (i + 1) ((i,j)::qs))
          handle Conflict =>if j = n then Conflict else try (j + 1)
    in try 1 end
  in addqueen 1 [ ] end
```


What we did today ...

- Continuations
  recursive arg. gets smaller / continuation grows bigger
- Exceptions
  define, raise, catch
  backtracking