(* so far, everything is pure... *)

val r = ref 0
val x = !r
val _ = r := 1
val y = !r

val bump = \x => x + 1
val ding = \x => x - 1
val s = ref bump
val x = (!s) 1
val _ = s := ding
val y = (!s) 1

val a = Array.array (4, "burrito")
(* arrays are indexed from 0 *)
val b = Array.sub (a, 2)
val _ = Array.update (a, 2, "taco")
(* what's the difference between "val" and "fun"? *)

val add = \x y => x + y
val concat = \x y => x ^ y

(* val countdown = fn n =>
   case n
     of 0 => []
     | n => n :: countdown (n - 1)
   *)
val f = ref (\x => x + 2)

val fact =
  \x =>
  if x ≤ 0
  then 1
  else x × (!f) (x - 1)
val _ = f := fact
(* aside from let, fun, and refs, anything else? *)

datatype T =
  T of (int × int × T) → (int × int × T)

val fact_aux =
  \(n, r, T f) =>
  if n ≤ 0
  then (n, r, T f)
  else f (n - 1, n × r, T f)
val fact' =
  \n =>
  #2 (fact_aux (n, 1, T fact_aux))