```
theory Output
  imports Base
begin

datatype 'a list = Nil | Cons 'a "'a list"

fun lookup :: "'a list ⇒ nat ⇒ 'a option" where
  "lookup Nil _ = None"
| "lookup (Cons x _) 0 = Some x"
| "lookup (Cons _ xs) (Suc n) = lookup xs n"

fun update :: "'a list ⇒ nat ⇒ 'a ⇒ 'a list" where
  "update (Cons _ xs) 0 y = Cons y xs"
| "update (Cons x xs) (Suc n) y = Cons x (update xs n y)"
| "update xs _ _ = xs"

partial_function (heap) example :: "nat ref ⇒ nat ref ⇒ nat Heap" where
  "example a b = do {
     a ← !a;
     b ← !b;
     return (a + b)
  }"

end
```