Left side:

```
01    type DiceVals   = [ Integer ]
02    type DiceTurn  = [(Bool, Integer)]
03
04    pop :: DiceTurn
05        -> Maybe ((Bool, Integer), DiceTurn)
06    pop [] = Nothing
07    pop (a:as) = Just (a, as)
08
09    allRolls :: DiceTurn
10             -> Integer
11             -> [ (DiceVals, Integer) ]
12    allRolls t n = [ (vals, n-1) |
13      vals <- allRollsNoN t ]
14
15    allRollsNoN :: DiceTurn -> [ DiceVals ]
16    allRollsNoN t = case pop t of
17
```

Right side:

```
01    import Data.List (uncons)
02
03    type DiceVals   = [ Integer ]
04    type DiceTurn  = [(Bool, Integer)]
05
06    allRolls :: DiceTurn
07             -> Integer
08             -> [ (DiceVals, Integer) ]
09    allRolls t n = [ (vals, n-1) |
10      vals <- allRollsNoN t ]
11
12    allRollsNoN :: DiceTurn -> [ DiceVals ]
13    allRollsNoN t = case uncons t of
14
```