

```

01 type DiceChoice = [ Bool ]
02 type DiceVals   = [ Integer ]
03 type DiceTurn   = (DiceChoice, DiceVals)
04
05 pop :: DiceTurn
06     -> Maybe ((Bool, Integer), DiceTurn)
07 pop ([], []) = Nothing
08 pop (chosen:choices, v:vs) = Just ((chosen, v),
09     (choices, vs))
10 pop (:_ , []) = error "Invariant violated: missing
11     val"
12 pop ([], _:_) = error "Invariant violated: missing
13     choice"
14
15 allRolls :: DiceTurn
16         -> Integer
17
18 example =
19     let diceChoices = [False, True, True, False, False]
20         diceVals = [ 6, 4, 4, 3, 1 ]
21     in mapM_ print $ allRolls (diceChoices, diceVals) 2
22

```

```

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
12 example =
13     let diceChoices = [False, True, True, False, False]
14         diceVals = [ 6, 4, 4, 3, 1 ]
15     in mapM_ print $ allRolls
16         (zip diceChoices diceVals) 2
17

```