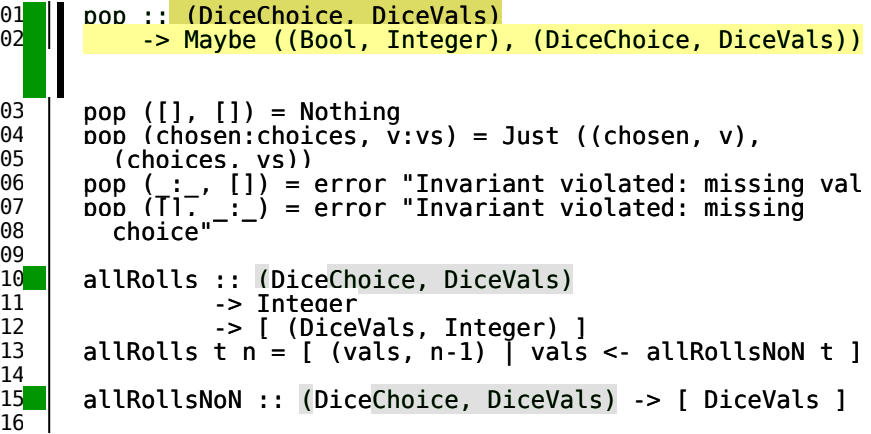


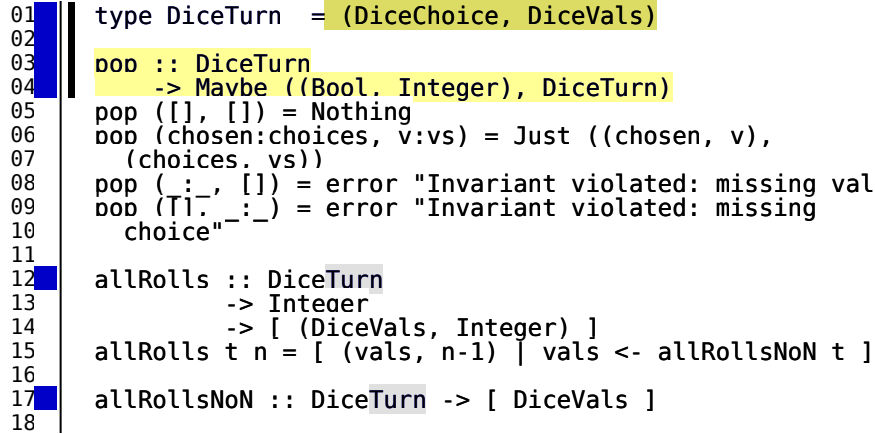
```
pop :: (DiceChoice, DiceVals)
    -> Maybe ((Bool, Integer), (DiceChoice, DiceVals))


pop ([], []) = Nothing
pop (chosen:choices, v:vs) = Just ((chosen, v),
  (choices, vs))
pop (_:_, []) = error "Invariant violated: missing val
pop ([], _:_) = error "Invariant violated: missing
  choice"

allRolls :: (DiceChoice, DiceVals)
         -> Integer
         -> [ (DiceVals, Integer) ]
allRolls t n = [ (vals, n-1) | vals <- allRollsNoN t ]

allRollsNoN :: (DiceChoice, DiceVals) -> [ DiceVals ]

```



```
type DiceTurn  = (DiceChoice, DiceVals)

pop :: DiceTurn
    -> Maybe ((Bool, Integer), DiceTurn)
pop ([], []) = Nothing
pop (chosen:choices, v:vs) = Just ((chosen, v),
  (choices, vs))
pop (_:_, []) = error "Invariant violated: missing val
pop ([], _:_) = error "Invariant violated: missing
  choice"

allRolls :: DiceTurn
         -> Integer
         -> [ (DiceVals, Integer) ]
allRolls t n = [ (vals, n-1) | vals <- allRollsNoN t ]

allRollsNoN :: DiceTurn -> [ DiceVals ]

```