

```

01 allRolls :: DiceChoice
02     -> (DiceVals, Integer)
03
04     -> [ (DiceVals, Integer) ]
05 allRolls choices (vs, n) = case pop choices vs of
06   Nothing -> [ ([], n-1) ]
07   Just ((chosen, v), (choices, vs)) ->
08     allRolls choices (vs, error "Didn't expect to use"
09       >>= \(roll, _) -> [ (d:roll, n-1)
10         | d <- rollList ]
11     where
12       rollList = if chosen then [v] else [ 1..6 ]
13
14 example =
15   let diceChoices = [ False, True, True, False, False
16     diceVals = [ 6, 4, 4, 3, 1 ]
17   in mapM_ print $ allRolls diceChoices (diceVals, 2)

```

```

01 allRolls :: DiceChoice
02     -> DiceVals
03     -> Integer
04
05     -> [ (DiceVals, Integer) ]
06 allRolls choices vs n = case pop choices vs of
07   Nothing -> [ ([], n-1) ]
08   Just ((chosen, v), (choices, vs)) ->
09     allRolls choices vs (error "Didn't expect to use"
10       >>= \(roll, _) -> [ (d:roll, n-1)
11         | d <- rollList ]
12     where
13       rollList = if chosen then [v] else [ 1..6 ]
14
15 example =
16   let diceChoices = [ False, True, True, False, False
17     diceVals = [ 6, 4, 4, 3, 1 ]
18   in mapM_ print $ allRolls diceChoices diceVals 2

```