```
allRolls :: (DiceChoice, DiceVals)
         -> Integer
         -> [ (DiceVals, Integer) ]
allRolls t n = case pop t of
  Nothing -> [ ([], n-1) ]
  Just ((chosen, v), t) ->
    allRolls t (error "Didn't expect to use")
      >>= \(roll, ) -> [ (d:roll,  n-1)
        | d <- rollList ]
        where
          rollList = if chosen then [v] else [ 1..6 ]

```

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

allRollsNoN :: (DiceChoice, DiceVals) -> [ DiceVals ]
allRollsNoN t = case pop t of
  Nothing -> [ [] ]
  Just ((chosen, v), t) ->
    allRollsNoN t >>=
      \roll -> [ d:roll | d <- rollList ]
        where
          rollList = if chosen then [v] else [ 1..6 ]

```