```
public class NQueen {
    var board: [String : Bool]
    var n: Int

    private var globalHelper: Int

    init() {
        n = 10
        board = [:]
        globalHelper = 0
    }

    public func solveFor(num: Int) -> Int {
        n = num
        globalHelper = 0
        initBoard()

        solve(row: 0)
        print(n)

        return globalHelper
    }

    private func initBoard() {

        for i in 0...n-1 {
            for j in 0...n-1 {
                board["\(i)|\(j)"] = false
            }
        }
    }
}

extension NQueen {
```

```
private func solve(row: Int) {
    if row >= n {
        globalHelper += 1
        return
    }
    for i in 0...n-1 {
        if isSafe(x: row, y: i) {
            board["\(row)|\(i)"] = true
            solve(row: row + 1)
            board["\(row)|\(i)"] = false
        } else {
            return
        }
    }

}

private func inBounds(x: Int, y: Int) -> Bool {
    return x >= 0 && y >= 0 && x < n && y < n
}

private func isSafe(x: Int, y: Int) -> Bool {

    if !inBounds(x: x, y: y) {
        return false
    }
    if !checkVertical(x: x, y: y) {
        return false
    }
    if !checkHorizontal(x: x, y: y) {
        return false
    }
    if !checkDiagonals(x: x, y: y) {
        return false
    }
```

```
    return true
}

private func checkVertical(x: Int, y: Int) -> Bool {
    for i in 0...n-1 {
        if inBounds(x: x, y: i) && board["\(x)|\(i)"] == true {
            return false
        }
    }
    return true
}

private func checkHorizontal(x: Int, y: Int) -> Bool {
    for i in 0...n-1 {
        if inBounds(x: x, y: y) && board["\(i)|\(y)"] == true {
            return false
        }
    }
    return true
}

private func checkDiagonals(x: Int, y: Int) -> Bool {
    var i: Int = 0
    var j: Int = 0
    // top left to bottom right
    while(inBounds(x: i, y: j)) {
        if board["\(i)|\(j)"] == true {
            return false
        }
        if board["\(i)|\(j)"] == true {
            return false
        }
        if board["\(i)|\(j)"] == true {
            return false
        }
```

```
            if board["\(i)|\(j)"] == true {
                return false
            }
            i = i + 1
            j = j + 1
        }

        return true
    }
}

let nQueen = NQueen()
nQueen.solveFor(num: 8)
nQueen.solveFor(num: 3)
```