

---

**Problem 2:** The n-queens is the problem of placing n queens on  $n \times n$  chessboard such that no two queens can attack each other. Given an integer n, return all distinct solutions to the n -queens puzzle. Each solution contains a distinct boards configuration of the queen's placement, where 'Q' and '.' indicate queen and empty space respectively.

```
def solveNQueens(n):  
    def backtrack(row, col_placement, result):  
        if row == n: # Base case: All queens have been placed  
            result.append(generateBoard(col_placement))  
        else:  
            for col in range(n):  
                if isValidPlacement(row, col, col_placement):  
                    col_placement.append(col)  
                    backtrack(row + 1, col_placement, result)  
                    col_placement.pop()
```

```
def isValidPlacement(row, col, col_placement):  
    for i in range(row):  
        if col == col_placement[i] or \  
            row - i == abs(col - col_placement[i]):  
            return False  
    return True
```

```
def generateBoard(col_placement):  
    board = []  
    for i in range(n):  
        row = ['.'] * n  
        row[col_placement[i]] = 'Q'
```

```
        board.append("".join(row))

    return board

    result = []

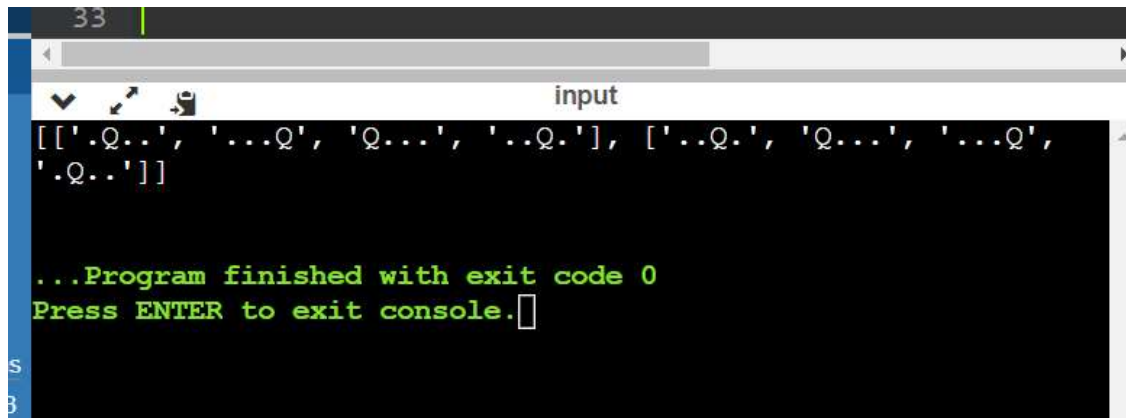
    backtrack(0, [], result)

    return result

n = 4

solutions = solveNQueens(n)

print(solutions)
```



```
33 |
input
[['.Q..', '...Q', 'Q...', '..Q.'], ['..Q.', 'Q...', '...Q',
'.Q..']]

...Program finished with exit code 0
Press ENTER to exit console.
```