Assignment no :- 5

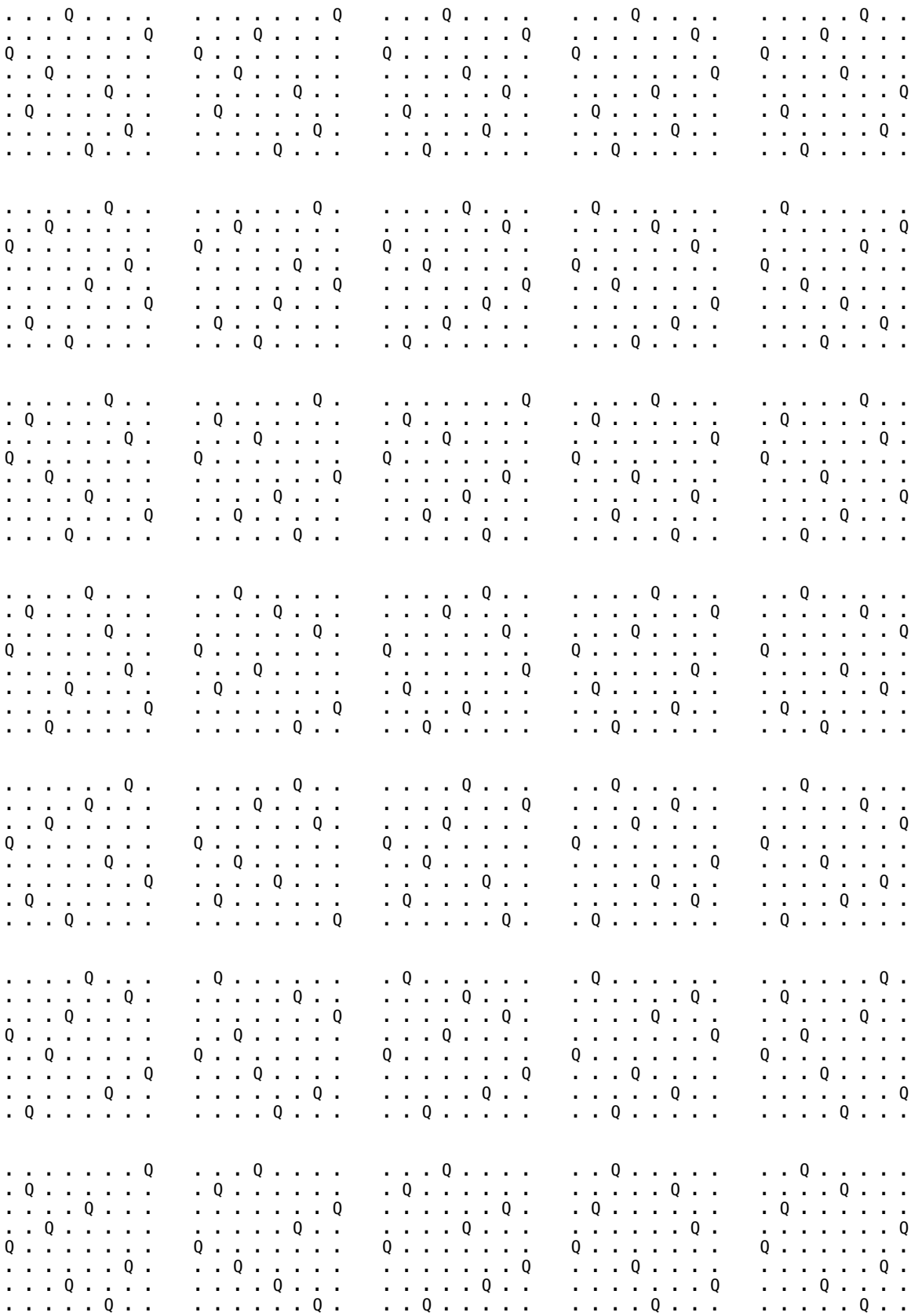
Program :-

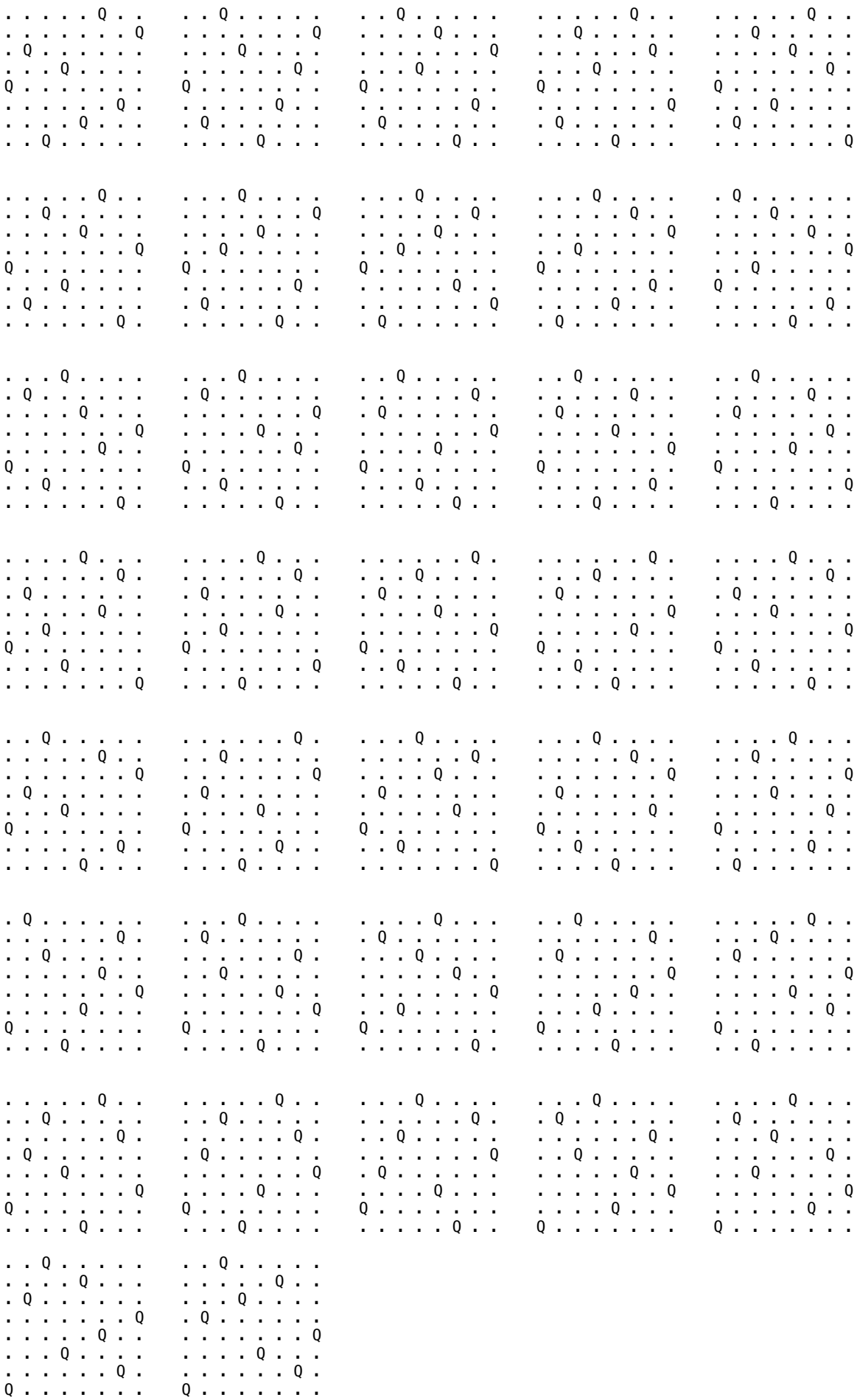
```
def formatSolution(board):
    return ["".join("Q " if cell else ". " for cell in row) for row in board]

def printMultipleSolutions(solutions, per_line=3):
    for i in range(0, len(solutions), per_line):
        batch = solutions[i:i+per_line]
        for row in range(len(batch[0])):  # N rows per board
            line = "   ".join(board[row] for board in batch)
            print(line)
        print("\n")

def solveNQueens_Backtracking(N):
    solutions = []

    def isSafeBT(board, row, col):
        for i in range(col):
            if board[row][i]:
                return False
        for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
            if board[i][j]:
                return False
        for i, j in zip(range(row, N), range(col, -1, -1)):
            if board[i][j]:
                return False
        return True

    def solveBT(board, col):
        if col == N:
            solutions.append(formatSolution(board))
            return True
        result = False
        for row in range(N):
            if isSafeBT(board, row, col):
                board[row][col] = 1
                result = solveBT(board, col + 1) or result
                board[row][col] = 0
        return result

    board = [[0] * N for _ in range(N)]
    solveBT(board, 0)
    printMultipleSolutions(solutions, per_line=4)

def solveNQueens_BranchAndBound(N):
    solutions = []
    board = [[0]*N for _ in range(N)]

    rowUsed = [False]*N
    slashCode = [[0]*N for _ in range(N)]
    backslashCode = [[0]*N for _ in range(N)]

    slashUsed = [False]*(2*N - 1)
    backslashUsed = [False]*(2*N - 1)

    for i in range(N):
        for j in range(N):
            slashCode[i][j] = i + j
            backslashCode[i][j] = i - j + (N - 1)

    def isSafeBB(i, j):
        return not (rowUsed[i] or slashUsed[slashCode[i][j]] or backslashUsed[backslashCode[i][j]])

    def solveBB(col):
        if col == N:
            solutions.append(formatSolution(board))
            return True
        result = False
        for row in range(N):
            if isSafeBB(row, col):
```

```
            board[row][col] = 1
            rowUsed[row] = True
            slashUsed[slashCode[row][col]] = True
            backslashUsed[backslashCode[row][col]] = True

            result = solveBB(col + 1) or result

            board[row][col] = 0
            rowUsed[row] = False
            slashUsed[slashCode[row][col]] = False
            backslashUsed[backslashCode[row][col]] = False

        return result

    solveBB(0)
    printMultipleSolutions(solutions, per_line=4)

# Run the solutions
N = 8
print("Backtracking Solution:\n")
solveNQueens_Backtracking(N)

print("\nBranch and Bound Solution:\n")
solveNQueens_BranchAndBound(N)
```

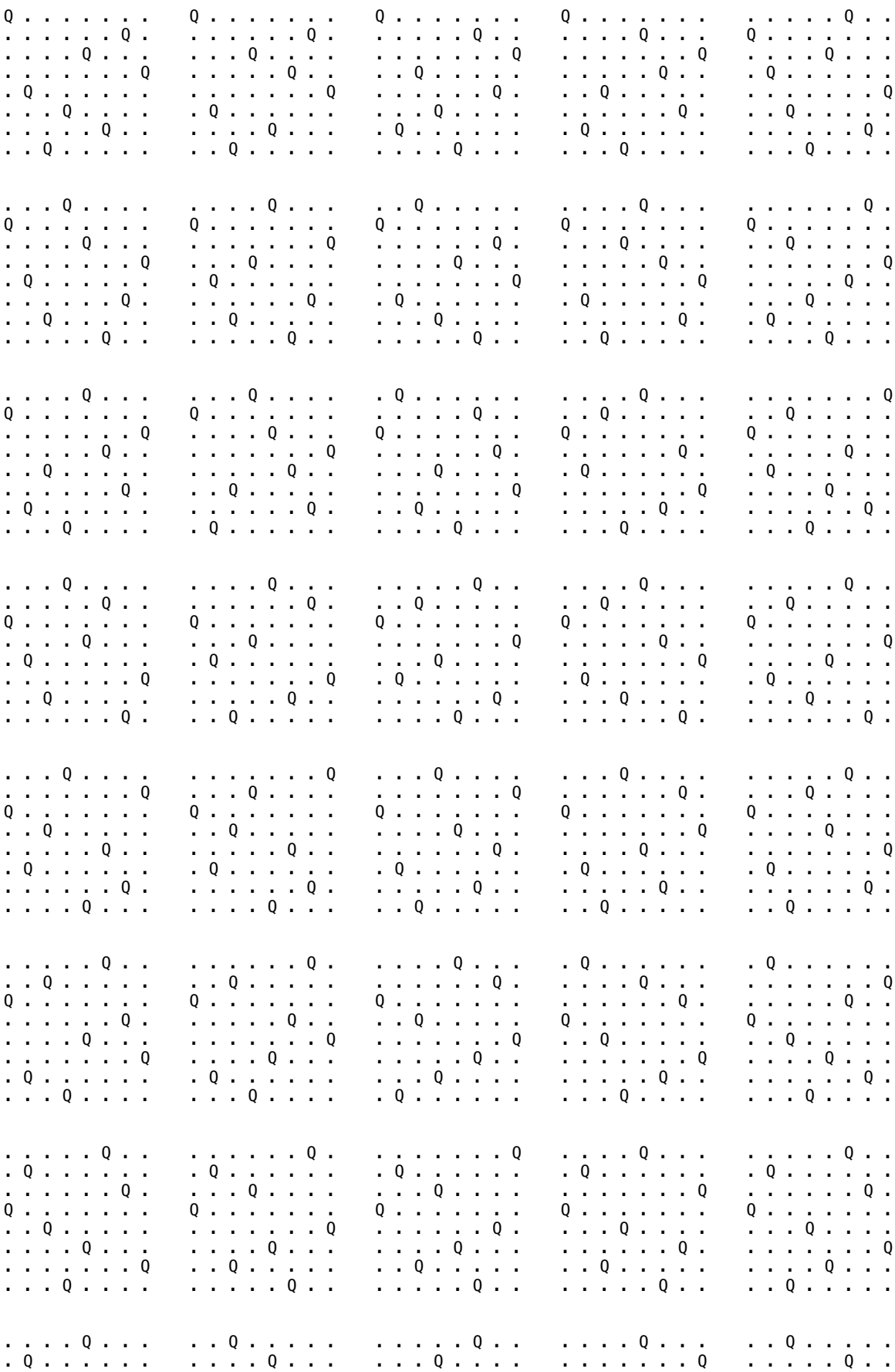
Output :-

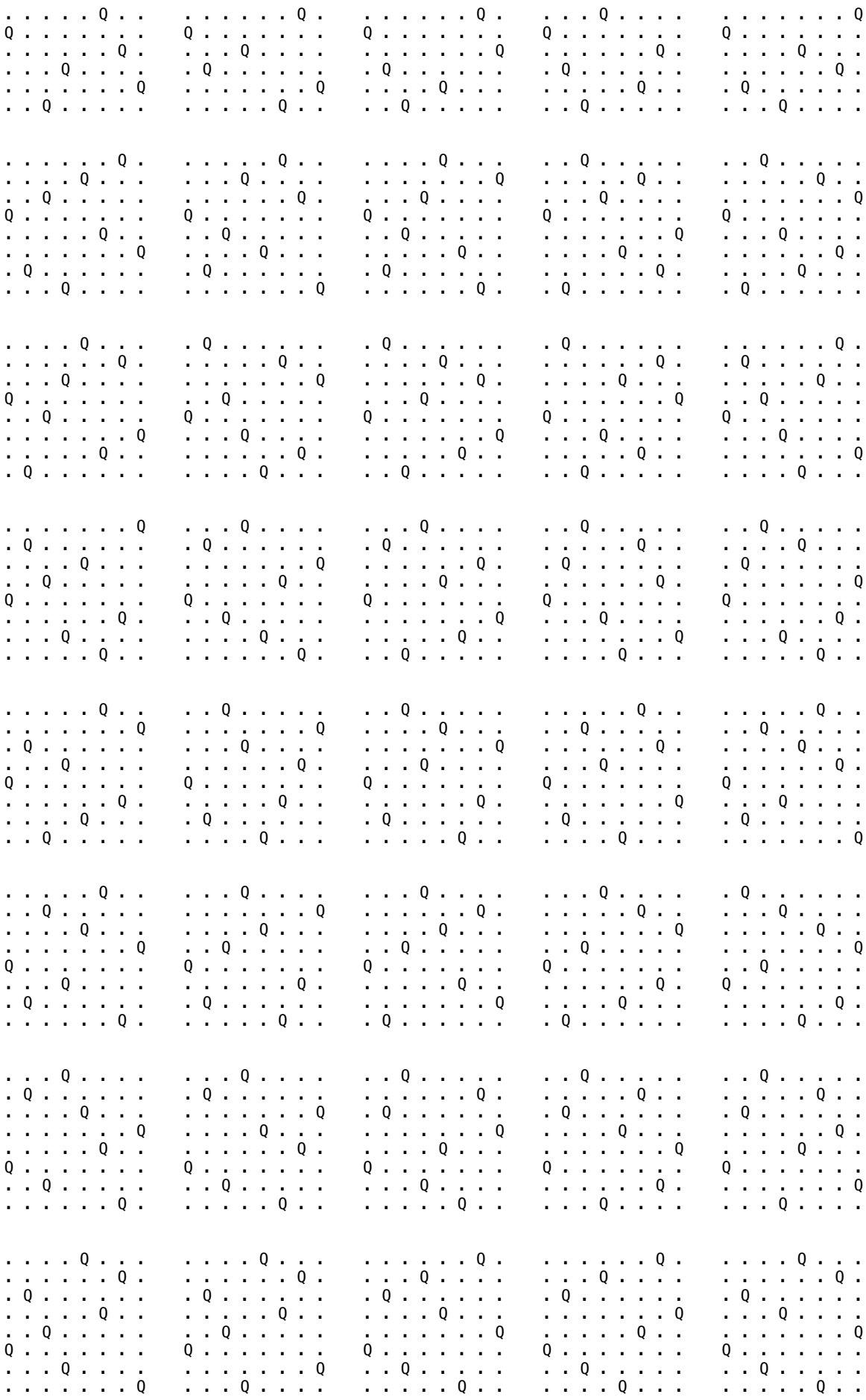
```
shivrajchaudar@Shivrajs-Macbook-Pro LP-II % python -u "/Users/Shivrajchaudar/Desktop/LP-II/A4.py"
Backtracking Solution:

Q . . . . . .    Q . . . . . .    Q . . . . . .    Q . . . . . .    . . . . . Q . .
. . . . . Q .    . . . . . Q .    . . . . . Q . .    . . . . Q . . .    Q . . . . . . .
. . . . Q . . .    . . . Q . . . .    . . . . . . . Q    . . . . . . . Q    . . . . Q . . .
. . . . . . . Q    . . . . . Q . .    . . Q . . . . .    . . . . . Q . .    . Q . . . . . .
. Q . . . . . .    . . . . . . . Q    . . . . . . Q .    . . Q . . . . .    . . . . . . . Q
. . . Q . . . .    . Q . . . . . .    . . . Q . . . .    . . . . . . Q .    . . Q . . . . .
. . . . . Q . .    . . . . Q . . .    . Q . . . . . .    . Q . . . . . .    . . . . . Q .
. . Q . . . . .    . . Q . . . . .    . . . . Q . . .    . . . Q . . . .    . . . Q . . . .

. . . Q . . . .    . . . . Q . . .    . . Q . . . . .    . . . . Q . . .    . . . . . . Q .
Q . . . . . .    Q . . . . . .    Q . . . . . .    Q . . . . . .    Q . . . . . . .
. . . . Q . . .    . . . . . . . Q    . . . . . . Q .    . . . Q . . . .    . . Q . . . . .
. . . . . . . Q    . . . Q . . . .    . . . . Q . . .    . . . . . Q . .    . . . . . . . Q
. Q . . . . . .    . Q . . . . . .    . . . . . . . Q    . . . . . . . Q    . . . . . Q . .
. . . . . . Q .    . . . . . Q . .    . Q . . . . . .    . Q . . . . . .    . . . Q . . . .
. . Q . . . . .    . . Q . . . . .    . . . Q . . . .    . . . . . . Q .    . Q . . . . . .
. . . . . Q . .    . . . . . Q . .    . . . . . Q . .    . . Q . . . . .    . . . . Q . . .

. . . . Q . . .    . . . Q . . . .    . Q . . . . . .    . . . . Q . . .    . . . . . . . Q
Q . . . . . .    Q . . . . . .    . . . . . Q . .    . . Q . . . . .    . . Q . . . . .
. . . . . . . Q    . . . . Q . . .    Q . . . . . .    Q . . . . . .    Q . . . . . . .
. . . . . Q . .    . . . . . . . Q    . . . Q . . . .    . . . . . . Q .    . . . . . Q . .
. . Q . . . . .    . . . . . Q . .    . . . . . . . Q    . Q . . . . . .    . Q . . . . . .
. . . . . . Q .    . . Q . . . . .    . . . . . . . Q    . . . . . . . Q    . . . . Q . . .
. Q . . . . . .    . . . . . . Q .    . . Q . . . . .    . . . . . Q . .    . . . . . . Q .
. . . Q . . . .    . Q . . . . . .    . . . . Q . . .    . . . Q . . . .    . . . Q . . . .

. . . Q . . . .    . . . . Q . . .    . . . . . Q . .    . . . . Q . . .    . . . . . Q . .
. . . . . Q . .    . . . . . . Q .    . . Q . . . . .    . . Q . . . . .    . . Q . . . . .
Q . . . . . .    Q . . . . . .    Q . . . . . .    Q . . . . . .    Q . . . . . .
. . . . Q . . .    . . . Q . . . .    . . . . . . . Q    . . . . . Q . .    . . . . . . . Q
. Q . . . . . .    . Q . . . . . .    . . . Q . . . .    . . . . . . . Q    . . . . Q . . .
. . . . . . . Q    . . . . . . . Q    . Q . . . . . .    . Q . . . . . .    . Q . . . . . .
. . Q . . . . .    . . . . . Q . .    . . . . . . Q .    . . . Q . . . .    . . . Q . . . .
. . . . . . Q .    . . Q . . . . .    . . . . Q . . .    . . . . . . Q .    . . . . . . Q .
```

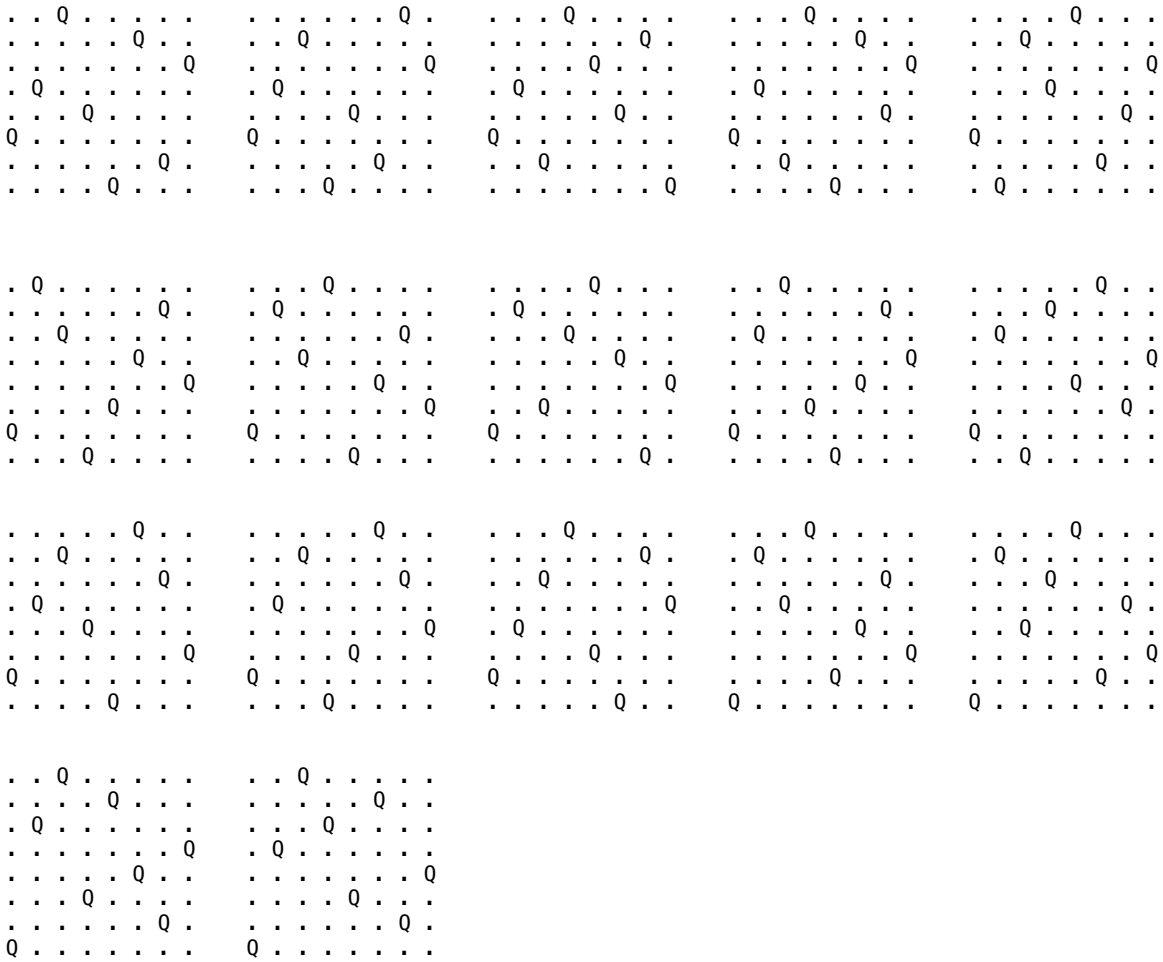




Branch and Bound Solution:







```
shivrajchaudar@Shivrajs-Macbook-Pro LP-II %
```