```python
import random

def calculate_attacks(board):
    attacks = 0
    n = len(board)
    for i in range(n):
        for j in range(i+1, n):
            if board[i] == board[j]:
                attacks += 1
            if abs(board[i] - board[j]) == abs(i - j):
                attacks += 1
    return attacks

def get_best_neighbor(board):
    n = len(board)
    current_attacks = calculate_attacks(board)
    best_board = list(board)
    best_attacks = current_attacks

    for col in range(n):
        original_row = board[col]
        for row in range(n):
            if row == original_row:
                continue
            new_board = list(board)
            new_board[col] = row
            attacks = calculate_attacks(new_board)
            if attacks < best_attacks:
                best_attacks = attacks
                best_board = new_board
    return best_board, best_attacks

def hill_climbing(n, max_restarts=100):
    board = [random.randint(0, n-1) for _ in range(n)]
    restarts = 0

    while restarts < max_restarts:
        current_attacks = calculate_attacks(board)
        if current_attacks == 0:
            return board

        next_board, next_attacks = get_best_neighbor(board)

        if next_attacks == current_attacks:
```

```
        next_board, next_attacks = get_best_neighbor(board)

        if next_attacks == current_attacks:
            board = [random.randint(0, n-1) for _ in range(n)]
            restarts += 1
        else:
            board = next_board

    return None

def print_board(board):
    if board is None:
        print("No solution found.")
        return
    n = len(board)
    for row in range(n):
        line = ""
        for col in range(n):
            if board[col] == row:
                line += "Q "
            else:
                line += ". "
        print(line)



n = 8
solution = hill_climbing(n)
print_board(solution)
```

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