Assignment 5

```
def is_safe(board, row, col, n):
    for i in range(col):
        if board[row][i] == 1:
            return False

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

    for i, j in zip(range(row, n), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def solve_n_queens(board, col, n):
    if col >= n:
        return True

    for i in range(n):
        if is_safe(board, i, col, n):
            board[i][col] = 1
            if solve_n_queens(board, col + 1, n):
                return True
            board[i][col] = 0

    return False
```

```
def print_board(board):
    for row in board:
        print(" ".join("Q" if cell == 1 else "." for cell in row))

def main():
    n = int(input("Enter the value of N: "))
    first_queen_col = int(input("Enter the column index (0-indexed) of the first queen: "))

    board = [[0 for _ in range(n)] for _ in range(n)]
    board[0][first_queen_col] = 1

    if solve_n_queens(board, 1, n):
        print("N-Queens matrix:")
        print_board(board)
    else:
        print("No solution exists for the given configuration.")

if __name__ == "__main__":
    main()
```

Output:

```
Enter the value of N: 6
Enter the column index (0-indexed) of the first queen: 4
N-Queens matrix:
. Q . . Q .
. . . . Q .
. . Q . . .
. . . . . Q
. . . Q . .
. . . . . .
```