ASSIGNMENT NO.5

INPUT:

```
import java.util.*;  // Import Scanner and List utilities

public class NQueen {

    // Function to check whether a queen can be safely placed at position (i, j)
    static boolean isAttack(int i, int j, int[][] board, int N) {

        // Check same column above
        for (int k = 0; k < i; k++)
            if (board[k][j] == 1)
                return true;

        // Check upper-left diagonal
        for (int k = i - 1, l = j - 1; k >= 0 && l >= 0; k--, l--)
            if (board[k][l] == 1)
                return true;

        // Check upper-right diagonal
        for (int k = i - 1, l = j + 1; k >= 0 && l < N; k--, l++)
            if (board[k][l] == 1)
                return true;

        // No conflict found
        return false;
    }

    // Recursive backtracking function to solve N-Queen problem
    static boolean NQueen(int row, int n, int N, int[][] board) {

        // Base case: all queens are placed
        if (n == 0)
            return true;

        // Try placing queen in each column of current row
        for (int j = 0; j < N; j++) {

            // Check if position (row, j) is safe
            if (!isAttack(row, j, board, N)) {

                // Place queen
                board[row][j] = 1;

                // Recurse to next row
                if (NQueen(row + 1, n - 1, N, board))
                    return true;

                // Backtrack if placing queen here doesn't lead to a solution
                board[row][j] = 0;
            }
        }
        return false; // No valid position found for this row
    }

    // Function to print the board
    static void printBoard(int[][] board, int N) {
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                System.out.print(board[i][j] + " ");
            }
            System.out.println();
        }
    }

    // Main function
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Input number of queens (size of board)
        System.out.print("Enter number of queens: ");
        int N = sc.nextInt();

        // Input column for the first queen in row 0
        System.out.print("Enter column (0 to " + (N - 1) + ") for the first queen in row 0:
");
        int firstCol = sc.nextInt();

        // Initialize chessboard with all zeros
        int[][] board = new int[N][N];

        // Place first queen in given column of row 0
        board[0][firstCol] = 1;

        // Try to solve the rest of the board
        if (NQueen(1, N - 1, N, board)) {
            System.out.println("Solution:");
            printBoard(board, N);
        } else {
            System.out.println("No solution exists starting from column " + firstCol + ".");
        }
        sc.close();
    }
}
```

OUTPUT:

```
Enter number of queens: 4
Enter column (0 to 3) for the first queen in row 0: 1
Solution:
0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0


=== Code Execution Successful ===

Enter number of queens: 5
Enter column (0 to 4) for the first queen in row 0: 3
Solution:
0 0 0 1 0
1 0 0 0 0
0 0 1 0 0
0 0 0 0 1
0 1 0 0 0


=== Code Execution Successful ===
```