LAB ASSIGNMENT – 05

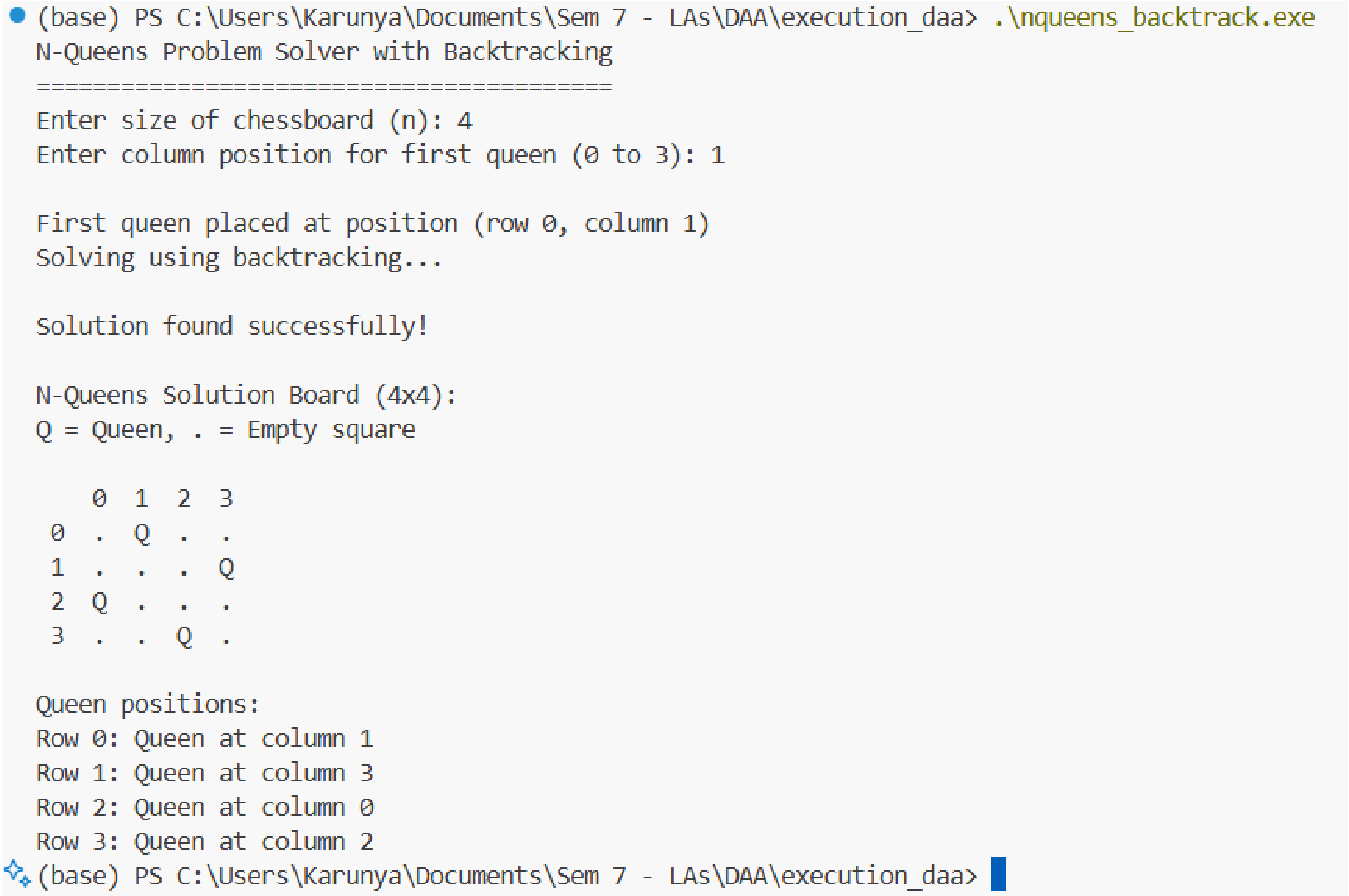
Program

```
#include <stdio.h>
#include <stdlib.h>

int n;
int *board; // board[i] stores column position of queen in row i

// Check if placing queen at (row, col) is safe
int isSafe(int row, int col) {
    for (int i = 0; i < row; i++) {
        // Check column and diagonal attacks
        if (board[i] == col ||
            abs(board[i] - col) == abs(i - row)) {
            return 0;
        }
    }
    return 1;
}

// Backtracking function to place remaining queens
int solveNQueens(int row) {
    if (row == n) {
        return 1; // All queens placed successfully
    }
    for (int col = 0; col < n; col++) {
        if (isSafe(row, col)) {
            board[row] = col;
            if (solveNQueens(row + 1)) {
                return 1; // Solution found
            }
        }
    }
    return 0; // No solution found
}
```

```
// Print the n-queens matrix
void printBoard() {
    printf("\nN-Queens Solution Board (%dx%d):\n", n, n);
    printf("Q = Queen, . = Empty square\n\n");

    // Print column numbers
    printf("   ");
    for (int j = 0; j < n; j++) {
        printf("%2d ", j);
    }
    printf("\n");
    // Print the board
    for (int i = 0; i < n; i++) {
        printf("%2d ", i); // Row number
        for (int j = 0; j < n; j++) {
            if (board[i] == j) {
                printf(" Q ");
            } else {
                printf(" . ");
            }
        }
        printf("\n");
    }
    printf("\nQueen positions:\n");
    for (int i = 0; i < n; i++) {
        printf("Row %d: Queen at column %d\n", i, board[i]);
    }
}

int main() {
    int first_queen_col;
    printf("N-Queens Problem Solver with Backtracking\n");
    printf("========================================\n");
    printf("Enter size of chessboard (n): ");
    scanf("%d", &n);
    if (n < 1) {
        printf("Invalid input! n must be positive.\n");
        return 1;
    }
    if (n == 2 || n == 3) {
        printf("No solution exists for n = %d\n", n);
        return 1;
    }
    printf("Enter column position for first queen (0 to %d): ", n-1);
    scanf("%d", &first_queen_col);
    if (first_queen_col < 0 || first_queen_col >= n) {
        printf("Invalid column position! Must be between 0 and %d\n", n-1);
        return 1;
    }
    // Allocate memory for board
    board = (int*)malloc(n * sizeof(int));
    // Place first queen
    board[0] = first_queen_col;
    printf("\nFirst queen placed at position (row 0, column %d)\n", first_queen_col);
    // Use backtracking to place remaining queens starting from row 1
    printf("Solving using backtracking...\n");
    if (solveNQueens(1)) {
        printf("\nSolution found successfully!\n");
        printBoard();
    }
    else {
        printf("\nNo solution exists with first queen at (row 0, column %d)\n", first_queen_col);
        // Show the initial state
        printf("\nInitial board state:\n");
        printf("   ");
        for (int j = 0; j < n; j++) {
            printf("%2d ", j);
        }
```

```
        printf("\n");
        printf(" 0 ");
        for (int j = 0; j < n; j++) {
            if (j == first_queen_col) {
                printf(" Q ");
            } else {
                printf(" . ");
            }
        }
        printf("\n");
    }

    free(board);
    return 0;
}
```

OUTPUT



```
(base) PS C:\Users\Karunya\Documents\Sem 7 - LAs\DAA\execution_daa> .\nqueens_backtrack.exe
N-Queens Problem Solver with Backtracking
========================================
Enter size of chessboard (n): 4
Enter column position for first queen (0 to 3): 1

First queen placed at position (row 0, column 1)
Solving using backtracking...

Solution found successfully!

N-Queens Solution Board (4x4):
Q = Queen, . = Empty square

    0  1  2  3
 0  .  Q  .  .
 1  .  .  .  Q
 2  Q  .  .  .
 3  .  .  Q  .

Queen positions:
Row 0: Queen at column 1
Row 1: Queen at column 3
Row 2: Queen at column 0
Row 3: Queen at column 2
(base) PS C:\Users\Karunya\Documents\Sem 7 - LAs\DAA\execution_daa>
```