********** 8 Queen's Problem using Backtracking **********

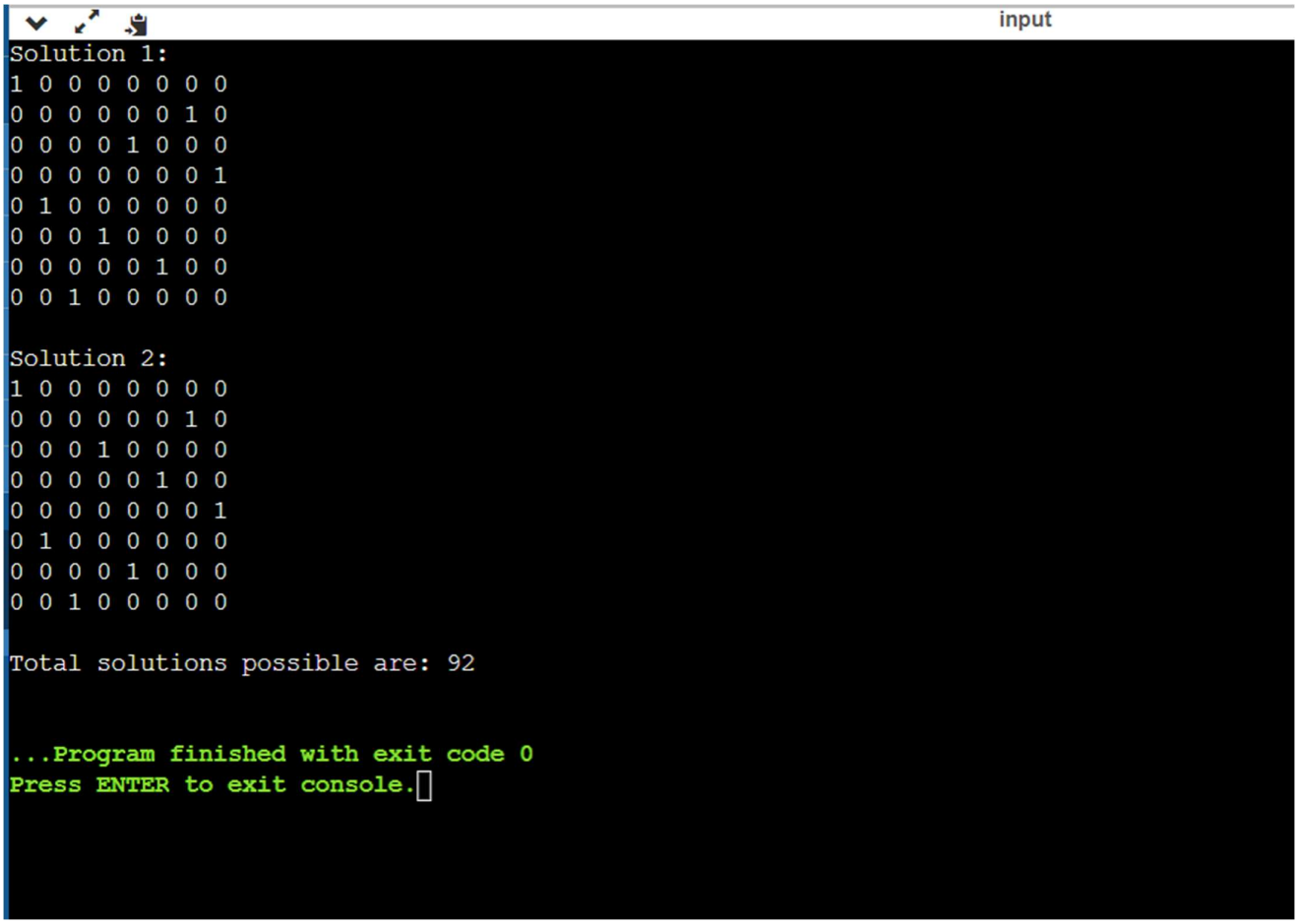
```
#include <stdio.h>
#include <stdbool.h>

#define N 8

void printBoard(int board[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            printf("%d ", board[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

bool isSafe(int board[N][N], int row, int col) {
    for (int i = 0; i < col; i++) {
        if (board[row][i]) {
            return false;
        }
    }

    for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {
        if (board[i][j]) {
            return false;
        }
    }

    for (int i = row, j = col; i < N && j >= 0; i++, j--) {
        if (board[i][j]) {
            return false;
        }
    }
```

```
    return true;
}

bool solveQueens(int board[N][N], int col) {
    if (col == N) {
        printBoard(board);
        return true;
    }

    bool foundSolution = false;
    for (int i = 0; i < N; i++) {
        if (isSafe(board, i, col)) {
            board[i][col] = 1;
            foundSolution = solveQueens(board, col + 1) || foundSolution;
            board[i][col] = 0;
        }
    }

    return foundSolution;
}

int main() {
    int board[N][N] = {0};

    if (solveQueens(board, 0) == false) {
        printf("No solution exists.\n");
    }

    return 0;
}
```

main.c

```c
#include <stdio.h>
#include <stdbool.h>

#define N 8

void printBoard(int board[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            printf("%d ", board[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

bool isSafe(int board[N][N], int row, int col) {
    for (int i = 0; i < col; i++) {
        if (board[row][i]) {
            return false;
        }
    }

    for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {
        if (board[i][j]) {
            return false;
        }
    }

    for (int i = row, j = col; i < N && j >= 0; i++, j--) {
        if (board[i][j]) {
            return false;
        }
    }

    return true;
}

bool solveQueens(int board[N][N], int col) {
    if (col == N) {
        printBoard(board);
        return true;
    }

    bool foundSolution = false;
    for (int i = 0; i < N; i++) {
        if (isSafe(board, i, col)) {
            board[i][col] = 1;
            foundSolution = solveQueens(board, col + 1) || foundSolution;
            board[i][col] = 0;
        }
    }

    return foundSolution;
}

int main() {
    int board[N][N] = {0};

    if (solveQueens(board, 0) == false) {
        printf("No solution exists.\n");
    }

    return 0;
}
```

OUTPUT


input
Solution 1:
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
Solution 2:
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0
Total solutions possible are: 92
...Program finished with exit code 0
Press ENTER to exit console.