

***** 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

```
1  #include <stdio.h>
2  #include <stdbool.h>
3
4  #define N 8
5
6  void printBoard(int board[N][N]) {
7      for (int i = 0; i < N; i++) {
8          for (int j = 0; j < N; j++) {
9              printf("%d ", board[i][j]);
10             }
11             printf("\n");
12         }
13         printf("\n");
14     }
15
16     bool isSafe(int board[N][N], int row, int col) {
17         for (int i = 0; i < col; i++) {
18             if (board[row][i]) {
19                 return false;
20             }
21         }
22
23         for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {
24             if (board[i][j]) {
25                 return false;
26             }
27         }
28
29         for (int i = row, j = col; i < N && j >= 0; i++, j--) {
30             if (board[i][j]) {
31                 return false;
32             }
33         }
34
35         return true;
36     }
37
38     bool solveQueens(int board[N][N], int col) {
39         if (col == N) {
40             printBoard(board);
41             return true;
42         }
43
44         bool foundSolution = false;
45         for (int i = 0; i < N; i++) {
46             if (isSafe(board, i, col)) {
47                 board[i][col] = 1;
48                 foundSolution = solveQueens(board, col + 1) || foundSolution;
49                 board[i][col] = 0;
50             }
51         }
52
53         return foundSolution;
54     }
55
56     int main() {
57         int board[N][N] = {0};
58
59         if (solveQueens(board, 0) == false) {
60             printf("No solution exists.\n");
61         }
62
63         return 0;
64     }
```

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.
```