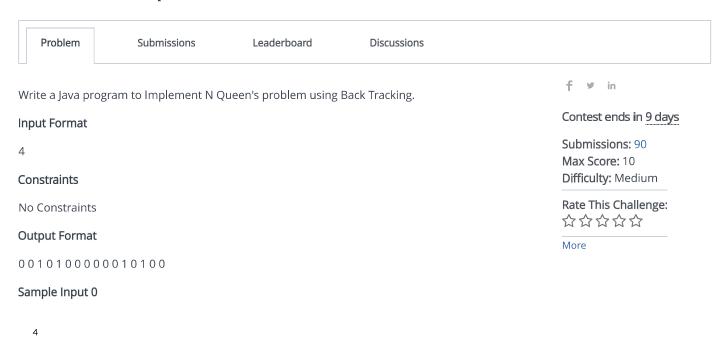
## N Queen's problem



Sample Output 0

```
Java 7
                                                                                                             *
 1 ▼import java.util.*;
 2 public class NQueenBacktracking
 3 ₹{
 4
        int n;
 5
        NQueenBacktracking(int n)
 6
 7
             this.n = n;
 8
 9
10
        /* Display solution*/
11
        void displaySolution(int queenBoard[][])
12 🔻
             for (int i = 0; i < n; i++)
13
14 ▼
                 for (int j = 0; j < n; j++)
    System.out.print(" " + queenBoard[i][j] + " ");</pre>
15
16 ▼
17
                     System.out.println();
18
             }
19
20
21 •
        /* isSafe() function check if a queen can be placed on queenBoard[row][col]. */
        boolean isSafe(int queenBoard[][], int row, int col)
22
23
             int i, j;
24
             /* for row on left side */
25
```

```
for (i = 0; i < col; i++)
26
27 1
                if (queenBoard[row][i] == 1)
28
                     return false;
             /* for upper diagonal on left side */
29
            for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
30
                if (queenBoard[i][j] == 1)
31 🔻
                     return false;
32
33 ▼
             /* for lower diagonal on left side */
            for (i = row, j = col; j >= 0 \&\& i < n; i++, j--)
34
35 ₹
                if (queenBoard[i][j] == 1)
36
                     return false;
37
            return true;
38
        }
39
        /* Utility function for N Queen problem solution */
40 •
        boolean utilityFunctionNQueen(int queenBoard[][], int col)
41
42 🔻
43 ▼
        /* base case when all queens are placed */
           if (col >= n)
44
45
               return true;
            /* for this column try placing this queen in all rows one by one */
46
47
            for (int i = 0; i < n; i++)
48
            /* Check is it safe at queenBoard[i][col] */
49 🔻
                if (isSafe(queenBoard, i, col))
50
51
                     /* Place this queen in board[i][col] */
52 ▼
53 ▼
                    queenBoard[i][col] = 1;
54
55 ₹
                     /* recurence to place rest of the queens */
                     if (utilityFunctionNQueen(queenBoard, col + 1) == true)
56
57
                         return true;
58
                    /* \ {\tt Backtrack:} \ {\tt If solution doesn't achieved then remove queen from queenBoard[i]}
59
    [col] */
                         queenBoard[i][col] = 0;
60
61
                }
62
63
            }
64
            /* If we cannot place queen in any row in this column col, then return false */
65 ▼
              return false;
66
67
        /\star uses solveNQUtil () to solve the problem. Note that there may be more than one
68
        /* solutions, this function prints one of the feasible solutions.*/
69
70
        boolean mainSolutionNQueen()
71 🔻
            int queenBoard[][] = new int[n][n];
72 •
73
            if (utilityFunctionNQueen(queenBoard, 0) == false)
74
                System.out.print("Solution does not exist");
75
76
                return false;
77
            displaySolution(queenBoard);
78
79
            return true;
80
        }
81
        // Driver main method
82
83
        public static void main(String args[])
84 1
        {
            int n;
85
            //System.out.print("Enter size of queen board i.e. N: ");
86
            Scanner sc = new Scanner(System.in);
87
            n = sc.nextInt();
88
            NQueenBacktracking queen = new NQueenBacktracking(n);
89
            queen.mainSolutionNQueen();
90
        }
91
92
   }
93
```

e Submit (stdin)  utput (stdo	
utput (stdo	tdout)
utput (stdo	
1 0	
1 0	
	a
	O Company of the Comp
0 0	9
0 1	1
0 0	9
ed Output	ut
) 1 0	9
0 0	9
0 1	1
0 0	9
0 0 0	9 1

Run Code Submit Code

<u>♣ Upload Code as File</u> Test against custom input