Name: Phadtare Shivendra Uday **Class:** BE Div: B

Roll No: COB434 **Batch:** B2

Assignment No:5

Design n-Queens matrix having first Queen placed.use backtracking to place remaining Queens to generate the final n-queens matrix.

CODE:

```
#include <iostream> using
namespace std;

#define N 4


void printSolution(int board[N][N]) { for
    (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) cout <<
            board[i][j] << " ";
        cout << endl;
    }
}


bool isSafe(int board[N][N], int row, int col) { int i, j;


      for (i = 0; i < col; i++) if
      (board[row][i]) return
                  false;


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


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

    return true;
}


bool solveNQUtil(int board[N][N], int col) { if (col
    >= N)
        return true;


    for (int i = 0; i < N; i++) {
```

```
        if (isSafe(board, i, col)) { board[i][col] = 1;


            if (solveNQUtil(board, col + 1)) return
                true;


            board[i][col] = 0;
        }
    }

    return false;
}

bool solveNQ() {
    int board[N][N] = { {0, 0, 0, 0},
                        {0, 0, 0, 0},
                        {0, 0, 0, 0},
                        {0, 0, 0, 0} };

    if (!solveNQUtil(board, 0)) {
        cout << "Solution does not exist" << endl; return false;
    }

    printSolution(board); return
    true;
}

int main() { solveNQ();
    return 0;
}
```

Output:

```
(base) sspm@sspm:~$ g++ daa5.cpp (base)
sspm@sspm:~$ ./a.out
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
```