****

Assignment No: 6

Title Name: Design 8-Queens matrix having first Queen placed. Use backtracking to place remaining Queens to generate the final 8-queen's matrix.

Name: Aniket Rajani

Class : BE **Div**: 1 **Batch**: A

Roll No: 405A008

****

Program:

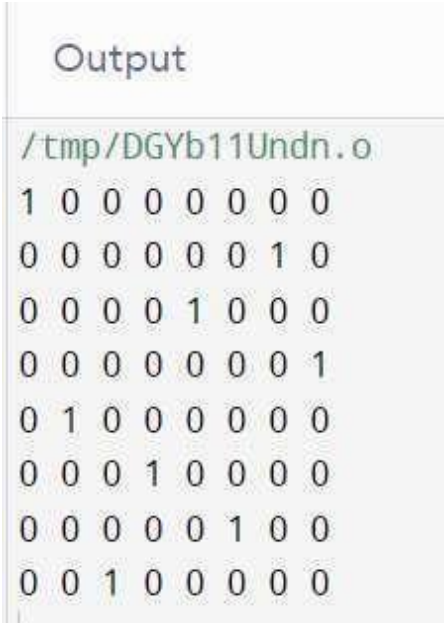
Code:

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#define N 8
using namespace std;

/* print solution */
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;
 }
}
   check if a queen can be placed on board[row][col]*/
bool isSafe(int board[N][N], int row, int col)
{
 int i, j;
 for (i = 0; i < col;
 {
    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;
}
/*solve N Queen problem */
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) == true)
      return true;
      board[i][col] = 0;

   }
 }
 return false;
}
/* solves the N Queen problem using Backtracking.*/
bool solveNQ()
{
 int board[N][N] = {0};
 if (solveNQUtil(board, 0) == false)
 {
   cout<<"Solution does not exist"<<endl;
   return false;
 }
 printSolution(board);
 return true;
}

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

}

Output:


Output
/tmp/DGYb11Undn.o
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