NAME : ROHIT SHEKHAR BAVISKAR
UID : 2021700004
BATCH : CSE DS D1
EXPT : DAA 7

AIM : To implement N Queens problem using backtracking

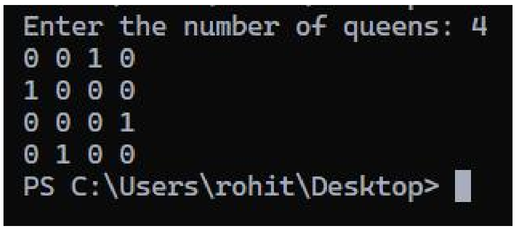
THEORY :
The goal of the N Queens problem is to arrange N queens on a NxN chessboard so that no two queens threaten one other. In other words, no two queens may be in the same row, column, or diagonal at the same time. Backtracking, a general algorithmic approach that includes systematically trying out different solutions and undoing those that don't work until a solution is discovered, can be used to solve the problem.

ALGORITHM :
```
1. Start in the leftmost column
2. If all queens are placed, return true
3. Try all rows in the current column. For each row:
a. If the queen can be placed safely in this row and column,
   mark this cell and recursively try to place the rest of the queens on the board
b. If the placement leads to a solution, return true
c. If the placement doesn't lead to a solution, unmark this cell and try the next row
4. If all rows have been tried and nothing worked, return false to trigger backtracking to the previous column
5. Repeat steps 3-4 for the previous column, trying the next row until a solution is found or all solutions have been tried
```

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

bool isSafe(int board[][10], int row, int col, int N) {
   int i, j;

   // Check this row on left side
   for (i = 0; i < col; i++)
      if (board[row][i])
         return false;

   // Check upper diagonal on left side
   for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
      if (board[i][j])
         return false;

   // Check lower diagonal on left side
```

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

    return true;
}

bool solveNQUtil(int board[][10], int col, int N) {
    if (col >= N)
        return true;

    for (int i = 0; i < N; i++) {
        if (isSafe(board, i, col, N)) {
            board[i][col] = 1;

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

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

    return false;
}

void printSolution(int board[][10], int N) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            printf("%d ", board[i][j]);
        }
        printf("\n");
    }
}

void solveNQ(int N) {
    int board[10][10] = {0};

    if (solveNQUtil(board, 0, N) == false) {
        printf("Solution does not exist");
        return;
    }

    printSolution(board, N);
}

int main() {
    int N;

    printf("Enter the number of queens: ");
    scanf("%d", &N);

    solveNQ(N);

    return 0;
}
```

OUTPUT :



```
Enter the number of queens: 4
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
PS C:\Users\rohit\Desktop>
```

CONCLUSION :

Successfully understood N Queens problem and its implementation using Backtracking in C