Name	Vedant Dapolikar
UID no.	2021700016
Experiment No.	7

AIM:	To implement N Queens problem
PROBLEM STATEMENT :	To implement N Queens problem using backtracking
ALGORITHM/ 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 The time complexity of the N Queens problem using backtracking is O(N!), where N is the size of the board. This is because there are N! possible ways to place N queens on an NxN board, and the backtracking algorithm tries all of them.

PROGRAM:

```c
#include <stdio.h>
#include <stdlib.h>

int n;

int check(int row, int col, int (*arr)[n])
{

   for (int i = 0; i < col; i++)
   {
       if (arr[row][i])
       {
           return 0;
       }
   }

   for (int i = row, j = col; i >= 0 && j >= 0; i--, j--)
   {
       if (arr[i][j])
       {
           return 0;
       }
   }

   for (int i = row, j = col; j >= 0 && i < n; i++, j--)
   {
       if (arr[i][j])
       {
           return 0;
       }
   }
   return 1;
}

int queens(int col, int (*arr)[n])
{
   if (col >= n)
   {
       return 1;
   }

   for (int i = 0; i < n; i++)
   {

       if (check(i, col, arr))
       {

           arr[i][col] = 1;

           if (queens(col + 1, arr))
           {
               return 1;
           }
```

```
            arr[i][col] = 0;
        }
    }

    return 0;
}

void printBoard(int (*arr)[n])
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("\t%d ", arr[i][j]);
        }
        printf("\n\n");
    }
}

int main()
{
    printf("No. of Queens(n): ");
    scanf("%d", &n);
    int arr[n][n];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            arr[i][j] = 0;
        }
    }
    if (queens(0, arr))
    {
        printf("\nSolution:\n");
        printBoard(arr);
    }
    else
    {
        printf("\nSolution doesn't exist.\n");
    }

    return 0;
}
```

RESULT:

```
 *  Executing task: /usr/bin/clang /Users/stephen03/Dev/repos/stepDAA/exp7/q2.c -o ../excs/q2 && ../excs/q2

No. of Queens(n): 4

Solution:
        0       0       1       0

        1       0       0       0

        0       0       0       1

        0       1       0       0

 *  Terminal will be reused by tasks, press any key to close it
```

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