NAME:	Eshan Bhuse
UID:	2021300013
SUBJECT	Design and analysis of algorithm
EXPERIMENTNO:	7
AIM:	**To implement Backtracking (N Queen Problem)**
ALGORITHM:	function solveNQueens(board, col, n): 1. if col >= n: print board return true 2. for row from 0 to n-1: 3. if isSafe(board, row, col, n): board[row][col] = 1 if solveNQueens(board, col+1, n): return true 4. board[row][col] = 0 5. return false function isSafe(board, row, col, n): 1. for i from 0 to col-1: 2. if board[row][i] == 1: 3. return false 4. for i,j from row-1, col-1 to 0, 0 by -1: 5.if board[i][j] == 1: return false 6. for i,j from row+1, col-1 to n-1, 0 by 1, 1: 7. if board[i][j] == 1: return false 8. return true 9. board = empty NxN chessboard

PROGRAM:

```c
#include <stdio.h>
#include <stdbool.h>
void printSolution(int n, int board[n][n])
{
 for (int i = 0; i < n; i++)
 {
     for (int j = 0; j < n; j++)
     {
         printf("%c ", board[i][j] ? 'Q' : '.');

     }
     printf("\n");

 }
 printf("\n");
}
bool isSafe(int n, 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])
```

```
48      if (board[i][j])
49      {
50          return false;
51
52      }
53
54  }
55  return true;
56  }
57  void solveNQueensUtil(int n, int board[n][n], int col)
58  {
59      if (col == n)
60      {
61          printSolution(n, board);
62          return;
63
64      }
65  for (int i = 0; i < n; i++)
66  {
67      if (isSafe(n, board, i, col))
68      {
69          board[i][col] = 1;
70          solveNQueensUtil(n, board, col+1);
71          board[i][col] = 0;
72
73      }
74
75  }
```

```
    }

}

}
void solveNQueens(int n)
{
    int board[n][n];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            board[i][j] = 0;

        }

    }
    solveNQueensUtil(n, board, 0);
}
int main()
{
    int n = 0;
    printf("\nEnter the dimension of the chessboard : ");
    scanf("%d",&n);
    solveNQueens(n);
    return 0;
}

```

```
Enter the dimension of the chessboard : 8
Q . . . . . . .
. . . . . . Q .
. . . . Q . . .
. . . . . . . Q
. Q . . . . . .
. . . Q . . . .
. . . . . Q . .
. . Q . . . . .

Q . . . . . . .
. . . . . . Q .
. . . Q . . . .
. . . . . Q . .
. . . . . . . Q
. Q . . . . . .
. . . . Q . . .
. . Q . . . . .

Q . . . . . . .
. . . . . Q . .
. . . . . . . Q
. . Q . . . . .
. . . . . . Q .
. . . Q . . . .
. Q . . . . . .
. . . . Q . . .

Q . . . . . . .
. . . . Q . . .
. . . . . . . Q
. . . . . Q . .
. . Q . . . . .
. . . . . . Q .
. Q . . . . . .
. . . Q . . . .

. . . . . Q . .
Q . . . . . . .
. . . . Q . . .
. Q . . . . . .
. . . . . . . Q
. . Q . . . . .
. . . . . . Q .
. . . Q . . . .
```

OBSERVATION:	The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other. Queens should not in a same row, same column and in a same diagonal. This problem can be solved using backtracking. In this problem, we place one queen in each row/column. If we see that the queen is under attack at its chosen position, we try the next position. If a queen is under attack at all the positions in a row, we backtrack and change the position of the queen placed prior to the current position.
CONCLUSION:	In this Practical, I performed the program of N queen problem using Backtracking. I learnt about the backtracking where we try to get all the solutions and then choose the best solution. Here, in our program also, if queen can not place in all the positions, we backtrack to previous one.