```
/*      The 4-Queens Problem
        In this code, '0' represents empty block while '1' represents block with a queen*/

#include<stdio.h>
#include<string.h>

int ctr = 0;                    //a counter variable for no. of solutions

int solve(int board[][4], int row, int cols[4], int ndiag[7], int rdiag[7])
{

        if(row==4)
        {
                int i,j;                //rows and columns of the 2d array Board
                printf("Possible solution no. %d: ",++ctr);
                for(i=0;i<4;i++)
                {
                        for(j=0;j<4;j++)
                        {
                                if(board[i][j]==1)
                                {
                                        printf("Q%d-(R%d,C%d) ",i+1,i+1,j+1);
                                }
                        }
                        printf("  ");
                }
                printf("\n\nRespective orientation of the Queens:\n");
                for(i=0;i<4;i++)
                {
                        for(j=0;j<4;j++)
                        {
```

```
                                printf("%d ",board[i][j]);
                        }
                        printf("\n");
                }
                printf("\n\n");
                return 0;
        }

        int col;
        for(col = 0;col<4;col++)
        {
                if(cols[col]==0 && ndiag[row+col]==0 && rdiag[row-col+3]==0)
                {
                        board[row][col] = 1;
                        cols[col] = 1;
                        ndiag[row+col] = 1;
                        rdiag[row-col+3] = 1;

                        solve(board,row+1,cols,ndiag,rdiag);

                        board[row][col] = 0;
                        cols[col] = 0;
                        ndiag[row+col] = 0;
                        rdiag[row-col+3] = 0;
                }
        }
}

int main()
{
        int n = 4;
```

```
    int board[4][4] = {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};   //represents the chess board

    int cols[4] = {0,0,0,0};
    //represents the columns occupied

    int ndiag[7] = {0,0,0,0,0,0,0};
    //represents the normal diagonals occupied

    int rdiag[7] = {0,0,0,0,0,0,0};
    //represents the reverse diagonals occupied


    printf("***4-Queens Problem***\n\n");

    solve(board,0,cols,ndiag,rdiag);

    return 0;

}
```

Result

CPU Time: 0.00 sec(s), Memory: 1424 kilobyte(s)

```
***4-Queens Problem***

Possible solution no. 1: Q1-(R1,C2)   Q2-(R2,C4)   Q3-(R3,C1)   Q4-(R4,C3)

Respective orientation of the Queens:
0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0


Possible solution no. 2: Q1-(R1,C3)   Q2-(R2,C1)   Q3-(R3,C4)   Q4-(R4,C2)

Respective orientation of the Queens:
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
```

TIME COMPLEXITY OF THE CODE : O(n^2)=O(4^2)

SPACE COMPLEXITY OF THE CODE : O(n^2) = O(4^2)

Here n = 4