```
#include <iostream>
#include <cstdlib>

using namespace std;

const int MAX = 10;

int SolnCount =0;

void fnChessBoardShow(int n, int row[MAX]);
bool fnCheckPlace(int KthQueen, int ColNum, int row[MAX]);
int NQueen(int k,int n, int row[MAX]);

/
*******************************************************************
**********
*Function    : main
*Input parameters: no parameters
*RETURNS     :   0 on success
*******************************************************************
**********/
int main(void)

{

    int n;
    int row[MAX];
    cout << "Enter the number of queens : ";
    cin >> n;
     if (!NQueen(0,n,row))
        cout << "No solution exists for the given problem instance."
<< endl;
    else
        cout << "Number of solution for the given problem instance
is : " << SolnCount << endl;
     return 0;
}
/
*******************************************************************
**********
*Function    : NQueen
*Description     : Function to place n queens on a nxn chess board
without any
*                   queen attacking any other queen
*Input parameters:
*   int k   -   kth queen
*   int n   - no of queens
*   int row[MAX] - vector containing column numbers of each queen
*RETURNS     : returns 1 if solution exists or zero otherwise
*******************************************************************
**********/

int NQueen(int k,int n, int row[MAX])
```

```
{
    static int flag;
    for(int i=0;i<n;i++)
    {
        if(fnCheckPlace(k,i,row) == true)
        {
            row[k] = i;
            if(k == n-1)
            {
                fnChessBoardShow(n,row);
                SolnCount++;
                flag = 1;
                return flag;
            }
            NQueen(k+1, n, row);
        }
    }
    return flag;
}
/
*********************************************************************
**********
*Function   : fnCheckPlace
*Description: Function to check whether a kth queen can be palced in
a specific
*                  column or not
*Input parameters:
*   int KthQueen    -   kth queen
*   int ColNum      - columnn number
*   int row[MAX]    - vector containing column numbers of each queen
*RETURNS    : returns true if the queen can be palced or false
otherwise
*********************************************************************
**********/
bool fnCheckPlace(int KthQueen, int ColNum, int row[MAX])

{

    for(int i=0; i<KthQueen; i++)
    {
        if(row[i] == ColNum || abs(row[i]-ColNum) == abs(i-
KthQueen))
            return false;
    }

    return true;
}

/
*********************************************************************
**********
*Function   : fnChessBoardShow
*Description: Function to graphically display solution to n queens
problem
```

```
*Input parameters:
*   int n   - no of queens
*   int row[MAX]    - vector containing column numbers of each queen
*RETURNS    : no value
*******************************************************************
**********/

void fnChessBoardShow(int n, int row[MAX])

{
    cout << "\nSolution #" << SolnCount+1 << endl << endl;

    for (int i=0; i<n; i++)
    {
        for (int j=0; j<n; j++)
        {
            if (j == row[i])
                cout << "Q ";
           else
                cout << "# ";
        }
        cout << endl;
    }
    cout << endl;

}
```

```
OUTPUT

SAMPLE 1

Enter the number of queens : 4

Solution #1

# Q # #

# # # Q

Q # # #

# # Q #

Solution #2

# # Q #

Q # # #

# # # Q

# Q # #

Number of solution for the given problem instance is : 2
```

SAMPLE 2

```
Enter the number of queens : 3 No solution exists for the given
problem instance.
```