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



bool isSafetoPlace(int board[][10],int row,int column,int n){
    /*Queen will be safe to place if and only if
        1) No Other Queen is Placed in same column -to find itterate over previous
        2) No Other Queen is Placed in Right Upper Diagonal
        3) No Other Queen is Placed in Left Upper Diagonal
    */
    // 1 - Same Column Check
    for(int i=0 ;i<row;i++){
        if(board[i][column]==1){
            /* Queen already placed in some previous row at the column we are checking for . Thus we cannot place so return false*/
            return false;
        }
    }



    // 2 - Right Upper Diagonal

        int x=row;
        int y=column;

        while(x>=0 && y<n){
            if(board[x][y]==1)
            {return false;}
            x--;
            y++;
        }

    //3 - Left Upper Diagonal

        x=row;
        y=column;

        while(x>=0 && y>=0){
            if(board[x][y]==1)
            {return false;}
            x--;
            y--;
        }
    /* if control reaches upto here it means , these conditions are not satisfied and Queen is safe to be placed*/

    return true;
}

void DisplayBoard(int board[][10],int n){
    for(int i=0;i<n;i++){

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

            if(board[i][j]==1){
                printf(" Q ");
            }
```

```
58
59              else {
60                  printf(" _ ");
61              }
62          }
63
64          printf("\n\n");
65      }
66  }
67
68  bool solveNQueen(int board[][10],int current_row,int n){
69      /*Base Case: successfully placed Queens in N rows */
70      if(current_row == n){
71          /* Print Board Config*/
72          printf("\n\n ************* \n\n");
73          DisplayBoard(board,n);
74          return false;
75      }
76
77      /* Recursie Case: Try to find the the right column in current row*/
78      for(int current_column=0;current_column<n;current_column++){
79
80          if(isSafetoPlace(board,current_row,current_column,n)){
81              // Place the Queen assuming this is correct position
82              board[current_row][current_column]=1;
83
84              /* Ask the Remaining Board if next Queen can be Placed in Next
    Row*/
85              bool canNextQueenBePlaced = solveNQueen(board,current_row+1,n);
86              if(canNextQueenBePlaced){
87                  /* it means we have placed Queen above is Right and need not
    be shifted*/
88                  return true;
89              }
90
91              /* if next queen cannot be placed , our above assumption is wrong
    and needs to be corrected , Queen needs to be shifted in the current row and
    possibiliites need to be recalculated*/
92              // Backtrack
93              else board[current_row][current_column]=0;
94          }
95
96      }
97
98      // At this control point we have tried all possible positions in current
    row but have failed to place a Queen, hence return False
99
100     return false;
101
102 }
103
104 int main(){
105
106     /* Initialize board to 0 */
107     int board[10][10]={0};
108
109
110     solveNQueen(board,0,4);
111
112
113 }
```