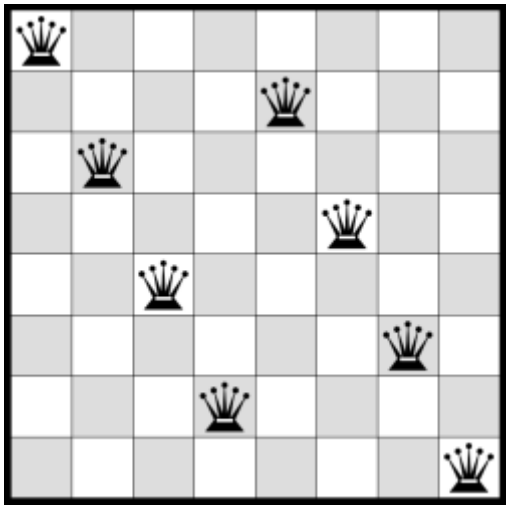
n – Queen Problem

Dr. Rahul Das Gupta

The eight queens puzzle is the problem of placing eight chess queens on an 8×8 chessboard so that no two queens threaten each other. Thus, a **solution** requires that no two queens share the same row, column, or diagonal.



/*

Let a Queen-1 at the cell (x1,y1). Now a new Queen-2 at the cell (x2,y2) is not safe if one of the following is true:

1. **y1=y2 (both the queens on the same horizontal line)**
2. **x1=x2 (both the queens on the same vertical line)**
3. **(i)If both the Queens at (x1, y1) and at (x2, y2) are on the diagonal at 45 degree [whose equation is y = x] then (x2-x1) = (y2-y1).**

   (ii)If both the Queens at (x1, y1) and at (x2, y2) are on the diagonal at 135 degree [whose equation is y = -x] then (x2-x1) = - (y2-y1).

   In general abs(x2-x1) = abs (y2-y1) (both the queens on the diagonal at an angle 45 degree or at 135 degree).

***/**

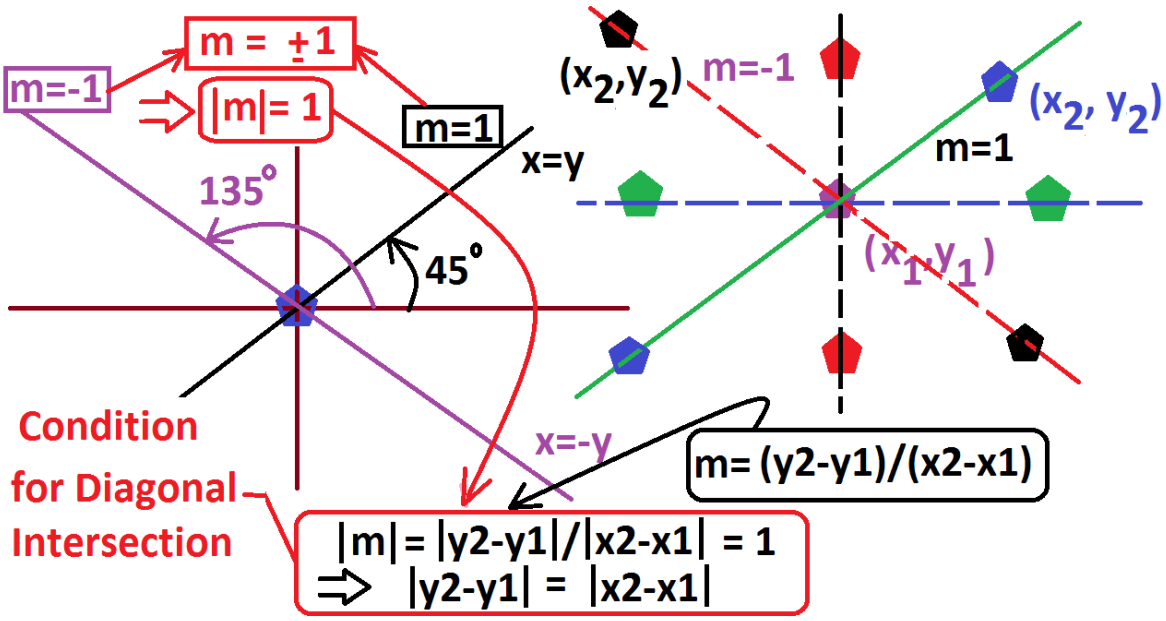


Fig.: Condition for Diagonal Intersection of Two Queens Placed at (x1, y1) and (x2, y2) respectively.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define MAX 8


typedef struct
{
 int *C;
 int no_queen;
}BOARD;

/*
==================================================
C[i] = column index of the ith Queen.
position of ith Queen : (i, C[i])
==================================================
*/


void initialisation(BOARD *, int );
void display_board(BOARD , int, int * );
int is_safe(BOARD , int , int );
void n_queen(BOARD * ,int, int, int * );
```

/*
==========
(x , y): coordinates of the new Queen.
int is_safe(BOARD B, int x, int y): To determine whether a Queen at **(x, y)** cell is safe.
Information in Hand: Column position of all the queens corresponding to row 0 to row (x-1) are known.
==========
*/

```
void initialisation(BOARD *B, int n)
{
  int i;
  B->no_queen = n;
  B->C = (int *) malloc(sizeof(int)*(n+1));
  for(i=1; i<=n; i++)
      B->C[i] = -1;
}

void display_board(BOARD B, int n, int *a_sol_no)
/*Displaying the Solution or Board Configuration.*/
{
  int i,j;
  printf("\n\n Solution %d ", ++(*a_sol_no));
  for(i=1;i<=n;i++)
  {
    printf("\n");
    for(j=1; j<=n ;j++)
      if(B.C[i] = = j)
        printf(" Q");
      else
        printf(" X");
   }
}
```

```
/* int  is_safe (BOARD B, int x, int y)  function returns 1 if a newly inserted
Queen is safe at (x, y) on the BOARD B, it will return 0 otherwise. */

int is_safe (BOARD B, int x, int y)
{
  int i;
  for(i=1; i<x ;i++) /*check all the (x-1) number of rows before x th row */
     if ( B.C[i] = = y || abs(x-i) = = abs(y-B.C[i]) )
     /* Position of two queens: (x, y) and (i, B.C[i]) */
          return 0;
  return 1;
}

void n_queen(BOARD *B, int k, int n, int *a_sol_no)
/*
================================================
C[i] = column index of the ith Queen.
Position of ith Queen : (i, C[i])
k = index of the current Queen to be inserted at k th row.
n = total no. of Queen to be placed.
================================================
*/
{
  int j;
  for(j=1; j<=n; j++) /* checking all the n columns one by one.*/
       if(is_safe(*B, k, j)) /* Is kth Queen at kth row is safe at jth column. */
         {
            B->C[k] = j; /* Place the kth Queen (at kth row) at jth column. */
           if(k = = n) /* All the n Queens are safely placed.*/
                display_board(*B, n, a_sol_no); /*Displaying the solution.*/
           else /* You have to place safely the next (k+1)th Queen.*/
                n_queen(B, k+1, n, a_sol_no); /* Place the (k+1)th Queen. */
         }
}
```

```
void main( )
{
 BOARD *p;
 int sol_no=0;
 p= (BOARD *) malloc(sizeof(BOARD));
 initialisation(p,8);
  n_queen(p, 1, 8, &sol_no);
}
```