N Queen Problem(N=4)

Initilize board and size

In [1]:

```
board = [[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]
N = 4
```

Method to check wheather the board is in safe state or not

In [2]:

```
def safeState(board,row,col):
    for i in range(N):
        if (board[row][i]==1 or board[i][col]):
            return False
    for i in range(N):
        for j in range(N):
            if( (i+j==row+col) or (i-j==row-col) ):
                if board[i][j]==1:
                    return False
    return True
```

Recursive funtion to place queens

- board : current status
- row : current row will go till N(i.e:4)

In [3]:

```
def solution(board,row):
    if(row>=N):
        return True #last+1 row -> all queens have been placed
    else:
        for col in range(N): #picking adj vertices
            if(safeState(board,row,col)): #if it's valid then we rec move down to next leve
                print("placed in {0},{1} //FORWARD EDGE".format(row+1,col+1))
                board[row][col]=1
                if(solution(board,row+1)):
                    return True
                print("Failed to place in {0},{1} //BACKTRACK:Level down".format(row+1,col+
                board[row][col]=0 #if solution is backtracked
            else:
                print("Failed to place in {0},{1} //BACKTRACK: move to adj".format(row+1,co
    return False
```

Used to display the board

In [4]:

```
def displayBoard():
    for i in range(N):
        for j in range(N):
            print (str(board[i][j]),end=" ")
        print()
```

Main function

- user input to place initial queen
- call solution function to solve the problem

In [5]:

```
n = int(input("Enter intial queen position:"))
board[0][n-1]=1
if(solution(board,1)):
    print("Solution:")
    displayBoard()
else:
    print("No solution possible")
```

```
Enter intial queen position:2
Failed to place in 2,1 //BACKTRACK: move to adj
Failed to place in 2,2 //BACKTRACK: move to adj
Failed to place in 2,3 //BACKTRACK: move to adj
placed in 2,4 //FORWARD EDGE
placed in 3,1 //FORWARD EDGE
Failed to place in 4,1 //BACKTRACK: move to adj
Failed to place in 4,2 //BACKTRACK: move to adj
placed in 4,3 //FORWARD EDGE
Solution:
0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0
```