

Exp NO: 1

N-Queen Problem

Date :

Aim :

To solve the N-Queen problem where the goal is to place n queens on a $n \times n$ chessboard such that no two queens attack each other.

Algorithm :

- Step 1: Start
- Step 2: Create a $n \times n$ chessboard with all cells set to 0, representing no queens placed.
- Step 3: Ensure no queen is in the same row, upper diagonal or lower diagonal for a given position.
- Step 4: Try placing a queen in each row of current column of it is safe using `isSafe()`.
- Step 5: Move to the next column if placing a queen works, else back track by removing queen.
- Step 6: If queen are placed in all columns return success.
- Step 7: Display the board.
- Step 8: If no solution exists print solution does not exist.

Program :

```
def is_safe(board, row, col, n):
```

```
    for i in range(col):
```

```
        if board[row][i] == 1:
```

```
            return False
```

```
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
```

```
        if board[i][j] == 1:
```

```
            return False
```

```
    return True
```

```
def solve Ngu ki (board, col, n):
```

```
    if col >= n:
```

```
        return True
```

```
    for i in range(n):
```

```
        if is_safe(board, i, col, n):
```

```
            board[i][col] = 1
```

```
            if solve Ngu ki (board, col+1, n) == True:
```

```
                return True
```

```
            board[i][col] = 0
```

```
    return False
```

```
def solve Ngu(n):
```

```
    board = [0]*n for i in range(n)
```

```
    if solve Ngu ki (board, 0, n) == False:
```

```
print ("solution does not exist")
```

```
return false
```

```
for i in board:
```

```
    print (i)
```

```
return true
```

```
n = int (input ("enter n value: "))
```

```
solve Nq(n)
```

Output :

enter n value : 5


[1, 0, 0, 0, 0]

[0, 0, 0, 1, 0]

[0, 1, 0, 0, 0]

[0, 0, 0, 0, 1]

[0, 0, 1, 0, 0]

Result 

Thus the n-queens problem program is executed & the output is verified successfully.