

Practical- 4 DAA

Name- Pratham Mahabare

Roll No.- 39

Code-

class NQueens:

```
def __init__(self) -> None:
```

```
    self.size = int(input("Enter size of chessboard: "))
```

```
    self.board = [[False]*self.size for _ in range(self.size)]
```

```
    self.count = 0
```

```
def printBoard(self):
```

```
    for row in self.board:
```

```
        for ele in row:
```

```
            if ele == True:
```

```
                print("Q",end=" ")
```

```
            else:
```

```
                print("X",end=" ")
```

```
        print()
```

```
    print()
```

```
def isSafe(self,row:int,col:int) -> bool:
```

```
    # Check Column(above and below of the (row,col))
```

```
    for i in self.board:
```

```
        if i[col] == True:
```

```
            return False
```

```
    # Check backward slash(\) diagonal only in above direction
```

```
    i = row
```

```
    j = col
```

```
    while i >= 0 and j >= 0:
```

```
        if self.board[i][j] == True:
```

```
            return False
```

```
        i -= 1
```

```
        j -= 1
```

```
    # Check backward slash(\) diagonal only in below direction
```

```
    i = row
```

```
    j = col
```

```
    while i < self.size and j < self.size:
```

```
        if self.board[i][j] == True:
```

```
            return False
```

```
        i += 1
```

```
        j += 1
```

```
    # Check forward slash diagonal(/) only in above direction
```

```
    i = row
```

```
    j = col
```

```

while i >= 0 and j < self.size:
    if self.board[i][j] == True:
        return False
    i -= 1
    j += 1

# Check forward slash diagonal(/) only in below direction
i = row
j = col
while i < self.size and j >= 0:
    if self.board[i][j] == True:
        return False
    i += 1
    j -= 1

```

```

return True

```

```

def set_position_first_queen(self):
    print("Enter coordinates of first queen: ")
    row = int(input(f"Enter row (1-{self.size}): "))
    col = int(input(f"Enter column (1-{self.size}): "))
    self.board[row-1][col-1] = True
    self.printBoard()

```

```

def solve(self,row:int):
    if row == self.size:
        self.count += 1
        self.printBoard()
        return

```

```

    if any(self.board[row]) is True:
        self.solve(row+1)
        return

```

```

    for col in range(self.size):
        if self.isSafe(row,col) == True:
            self.board[row][col] = True
            self.solve(row+1)
            self.board[row][col] = False

```

```

def displayMessage(self):
    if self.count > 0:
        print("Solution exists for the given position of the queen.")
    else:
        print("Solution doesn't exist for the given position of the queen.")

```

```

solver = NQueens()
solver.set_position_first_queen()
solver.solve(0)

```

solver.displayMessage()

Output-

```
pratham_mahabare@fedora:~/Downloads/LP-3/DAA$ python3 4_n-queens.py
Enter size of chessboard: 8
Enter coordinates of first queen:
Enter row (1-8): 1
Enter column (1-8): 1
Q X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X

Q X X X X X X X
X X X X Q X X X
X X X X X X X Q
X X X X X Q X X
X X Q X X X X X
X X X X X X Q X
X Q X X X X X X
X X X Q X X X X

Q X X X X X X X
X X X X X Q X X
X X X X X X X Q
X X Q X X X X X
X X X X X X Q X
X X X Q X X X X
X Q X X X X X X
X X X X Q X X X

Q X X X X X X X
X X X X X X Q X
X X X X Q X X X
X X X X X X X Q
X Q X X X X X X
X X X Q X X X X
X X X X X Q X X
X X Q X X X X X

Solution exists for the given position of the queen.
pratham_mahabare@fedora:~/Downloads/LP-3/DAA$
```