TASK:8

Implementation of **N-queen problem using backtracking algorithm** using prolog In the 4 Queens problem the object is to place 4 queens on a chessboard in such a way that no queens can capture a piece.

Aim: To Implement N-Queen's problem by using backtracking algorithm using python

Algorithm:

Step 1: k=queen and I is column number in which queen k is placed

Step 2: where x[] is a global array whose first k-1 values have been set

Step 3: Queen-place (k, i) returns true if a queen can be placed in the kth row and ith column otherwise return false

Step 4:ABS (r) returns the absolute value of r.

Step 5: for j<-1 to k-1 do if x[j]=1 or ABS(x[j]-1)= ABS (j-k) then return false

Step 6:for i<-1 to n do if Queen-place (k,i) then x[k] <- i if k=n then write

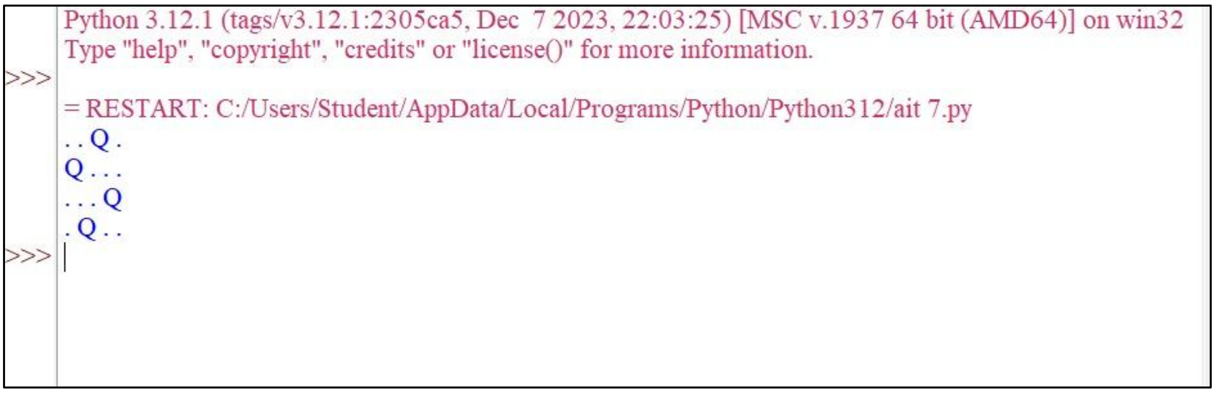
(x[i---n]) else N-Queen (k+1,n).

Program:

```
# Python3 program to solve N Queen

# Problem using backtracking global

N N = 4 def printSolution(board):

 for i  in  range(N):     for j  in  range(N):       if

board[i][j] == 1:    print("Q",end=" ")

                        else:

                                print(".",end=" ")

print()

def isSafe(board, row, col):

 # Check this row on left side  for i in

range(col):

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

                        return False
```

```
    # Check upper diagonal on left side
for i, j in zip(range(row, -1, -1),
                    range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    # Check lower diagonal on left side
for i, j in zip(range(row, N, 1),
                    range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    return True
def solveNQUtil(board, col):
    # Base case: If all queens are placed
# then return true        if col >= N:
        return True
 # Consider this column and try placing  # this
queen in all rows one by one  for i in range(N):
if isSafe(board, i, col):
            # Place this queen in board[i][col]
            board[i][col] = 1
            if solveNQUtil(board, col + 1) == True:
                return True
            board[i][col] = 0
    return False

def solveNQ():

    board = [[0, 0, 0, 0],

            [0, 0, 0, 0],

            [0, 0, 0, 0],

        [0, 0, 0, 0]]

    if solveNQUtil(board, 0) == False:

print("Solution does not exist")

        return False

    printSolution(board)

return True

# Driver Code if __name__

== '__main__':

    solveNQ()
```

Output:



```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Student/AppData/Local/Programs/Python/Python312/ait 7.py
. . Q .
Q . . .
. . . Q
. Q . .
>>> |
```

Result:

Thus the Implementation of N-queen problem using backtracking algorithm using Python was successfully executed and output was verified.