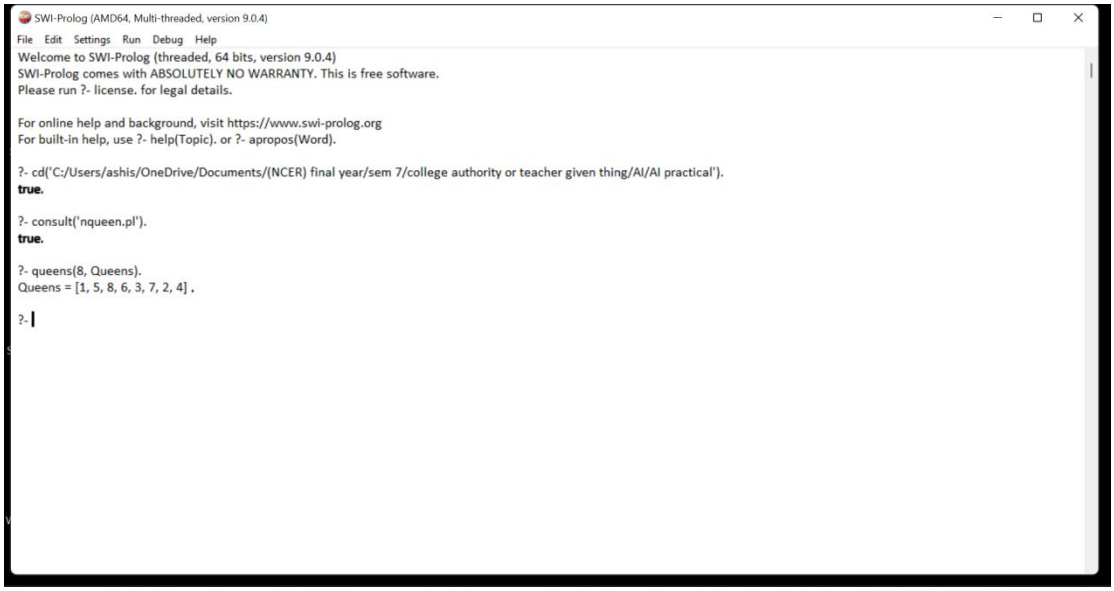
Q.1.Answer:

```
%%    queens(+N, -Queens) is nondet.
%
%    @param   Queens is a list of column numbers for placing the queens.

queens(N, Queens) :-
   length(Queens, N),
   board(Queens, Board, 0, N, _, _),
   queens(Board, 0, Queens).

board([], [], N, N, _, _).
board([_|Queens], [Col-Vars|Board], Col0, N, [_|VR], VC) :-
   Col is Col0+1,
   functor(Vars, f, N),
   constraints(N, Vars, VR, VC),
   board(Queens, Board, Col, N, VR, [_|VC]).

constraints(0, _, _, _) :- !.
constraints(N, Row, [R|Rs], [C|Cs]) :-
   arg(N, Row, R-C),
   M is N-1,
   constraints(M, Row, Rs, Cs).

queens([], _, []).
queens([C|Cs], Row0, [Col|Solution]) :-
   Row is Row0+1,
   select(Col-Vars, [C|Cs], Board),
   arg(Row, Vars, Row-Row),
   queens(Board, Row, Solution).

/** <examples>

?- queens(8, Queens).

*/
```

Q.1.Output:

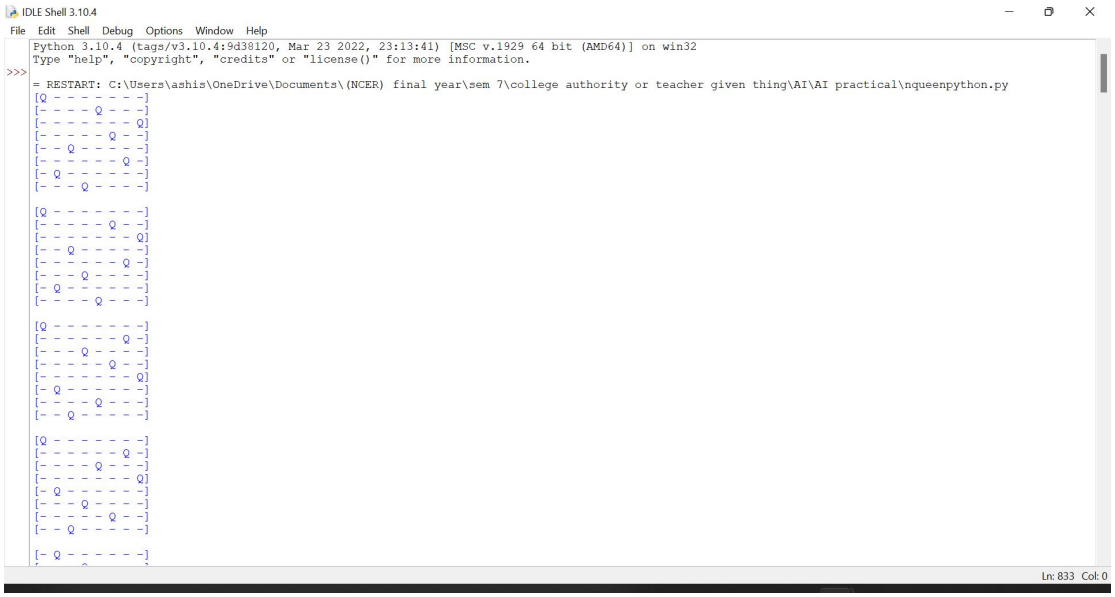


Q.2.Answer:

```
def isSafe(mat, r, c):

    # return false if two queens share the same column
    for i in range(r):
        if mat[i][c] == 'Q':
            return False

    # return false if two queens share the same `` diagonal
    (i, j) = (r, c)
    while i >= 0 and j >= 0:
        if mat[i][j] == 'Q':
            return False
        i = i - 1
        j = j - 1

    # return false if two queens share the same `/` diagonal
    (i, j) = (r, c)
    while i >= 0 and j < len(mat):
        if mat[i][j] == 'Q':
            return False
        i = i - 1
        j = j + 1

    return True


def printSolution(mat):
    for r in mat:
        print(str(r).replace(',', '').replace('\'', ''))
    print()
```

```
def nQueen(mat, r):

    # if `N` queens are placed successfully, print the solution
    if r == len(mat):
        printSolution(mat)
        return

    # place queen at every square in the current row `r`
    # and recur for each valid movement
    for i in range(len(mat)):

        # if no two queens threaten each other
        if isSafe(mat, r, i):
            # place queen on the current square
            mat[r][i] = 'Q'

            # recur for the next row
            nQueen(mat, r + 1)

            # backtrack and remove the queen from the current square
            mat[r][i] = '–'


if __name__ == '__main__':

    # `N × N` chessboard
    N = 8

    # `mat[][]` keeps track of the position of queens in
    # the current configuration
    mat = [['–' for x in range(N)] for y in range(N)]

    nQueen(mat, 0)
```

Q.2.Output:


IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\ashis\OneDrive\Documents\(NCER) final year\sem 7\college authority or teacher given thing\AI\AI practical\nqueenpython.py
[Q - - - - - - -]
[- - - - Q - - -]
[- - - - - - - Q]
[- - - - - Q - -]
[- - Q - - - - -]
[- - - - - - Q -]
[- Q - - - - - -]
[- - - Q - - - -]
[Q - - - - - - -]
[- - - - - Q - -]
[- - - - - - - Q]
[- - Q - - - - -]
[- - - - - - Q -]
[- - - Q - - - -]
[- Q - - - - - -]
[- - - - Q - - -]
[Q - - - - - - -]
[- - - - - - Q -]
[- - - Q - - - -]
[- - - - Q - - -]
[- - - - - - - Q]
[- Q - - - - - -]
[- - - - Q - - -]
[- - Q - - - - -]
[Q - - - - - - -]
[- - - - - - Q -]
[- - - - Q - - -]
[- - - - - - - Q]
[- Q - - - - - -]
[- - - Q - - - -]
[- - - - - Q - -]
[- - Q - - - - -]
[- Q - - - - - -]
Ln: 833 Col: 0