def is\_safe(board, row, col):

for i in range(row):

if board[i] == col or board[i] - i == col - row or board[i] + i == col + row:

return False

return True

def solve\_nqueens(board, row, n, solutions):

if row == n:

solutions.append(board[:])

return

for col in range(n):

if is\_safe(board, row, col):

board[row] = col

solve\_nqueens(board, row + 1, n, solutions)

def solve\_nqueens\_main(n):

board = [-1] \* n

solutions = []

solve\_nqueens(board, 0, n, solutions)

return solutions

def main():

n = int(input("Enter the number of queens (N): "))

solutions = solve\_nqueens\_main(n)

if not solutions:

print("No solutions found.")

else:

print("Total solutions:", len(solutions))

for count, solution in enumerate(solutions, start=1):

print("Solution", count, ":")

for row in solution:

print("".join("Q" if col == row else "." for col in range(n)))

print()

if \_\_name\_\_ == "\_\_main\_\_":

main()

output:-

Enter the number of queens (N): 4

Total solutions: 2

Solution 1 :

.Q..

...Q

Q...

..Q.

Solution 2 :

..Q.

Q...

...Q

.Q..