**Assignment 5**

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

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

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

return False

return True

def solve\_n\_queens(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\_n\_queens(board, col + 1, n):

return True

board[i][col] = 0

return False

def print\_board(board):

for row in board:

print(" ".join("Q" if cell == 1 else "0" for cell in row))

print("\n")

print("Matrix is:\n")

for row in board:

print(" ".join("1" if cell == 1 else "0" for cell in row))

def main():

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

board = [[0 for \_ in range(n)] for \_ in range(n)]

if solve\_n\_queens(board, 0, n):

print("N-Queens matrix:")

print\_board(board)

else:

print("No solution exists for the given configuration.")

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

main()

**Output:**

Enter the value of N: 8

N-Queens matrix:

Q 0 0 0 0 0 0 0

0 0 0 0 0 0 Q 0

0 0 0 0 Q 0 0 0

0 0 0 0 0 0 0 Q

0 Q 0 0 0 0 0 0

0 0 0 Q 0 0 0 0

0 0 0 0 0 Q 0 0

0 0 Q 0 0 0 0 0

Matrix is:

1 0 0 0 0 0 0 0

0 0 0 0 0 0 1 0

0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 1

0 1 0 0 0 0 0 0

0 0 0 1 0 0 0 0

0 0 0 0 0 1 0 0

0 0 1 0 0 0 0 0