**N QUEENS**

while True:

user\_input = input("Enter the n value of n\*n board: ")

try:

n = int(user\_input)

if n <= 0:

print("Please enter a positive number.")

continue

elif n == 3 or n==2:

print("Not possible. N-queen problem is only solvable for n > 3.")

continue

elif n >= 20:

print("n value should be lessthan 20")

continue

break

except ValueError:

print("Invalid input. Please enter a valid number.")

matrix = [[0 for i in range(n)] for j in range(n)]

while True:

try:

first\_row = int(input("Enter a row for placing a queen to start: "))

if first\_row <= 0:

raise ValueError()

continue

if first\_row > n:

raise ValueError()

continue

break

except ValueError:

print("Invalid input. Please enter valid integer")

while True:

try:

first\_col = int(input("Enter a col for placing a queen to start: "))

if first\_col <= 0:

raise ValueError()

continue

if first\_col > n:

raise ValueError()

continue

break

except ValueError:

print("Invalid input. Please enter valid integer")

matrix[first\_row - 1][first\_col - 1] = 1

6

def is\_safe(matrix, row, col, n):

for i in range(row):

if matrix[i][col] == 1:

return False

i, j = row, col

while i >= 0 and j >= 0:

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

return False

i -= 1

j -= 1

i, j = row, col

while j < n and i < n:

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

return False

i += 1

j += 1

i, j = row, col

while i >= 0 and j < n:

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

return False

i -= 1

j += 1

i, j = row, col

while j >= 0 and i < n:

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

return False

i += 1

j -= 1

return True

def solve\_nqueens(matrix, row, n,col\_taken):

if row >= n:

return True

if row == first\_row - 1:

return solve\_nqueens(matrix, row + 1, n,cols\_taken)

for col in range(n):

if col in cols\_taken:

continue

if is\_safe(matrix, row, col, n):

matrix[row][col] = 1

cols\_taken.add(col)

if solve\_nqueens(matrix, row + 1, n, cols\_taken):

return True

matrix[row][col] = 0

cols\_taken.remove(col)

return False

def print\_matrix(matrix):

for row in matrix:

print(" ".join(map(str, row)))

cols\_taken = {first\_col - 1}

if solve\_nqueens(matrix, 0, n,cols\_taken):

print("Solution found:")

print\_matrix(matrix)

else:

print("No solution exists.")