print("\n###CSE422\_Lab01\_Mohammad Shafkat Hasan\_19101077###\n")

# Task 01

def covid\_tracer(file\_name):

# file input

with open(file\_name, 'r') as file:

matrix = [[ele for ele in line.split(' ')] for line in file]

# removing line break(\n)

for index in range(len(matrix)):

matrix[index][-1] = matrix[index][-1].replace('\n', '')

# creating a tuple list of the positions of infect

infect = []

for row in range(len(matrix)):

for col in range(len(matrix[row])):

if 'Y' in matrix[row][col]:

infect.append((row, col))

counter = [] # counter to keep track of visited tuples or position within matrix

# appliying DFS to all areas in matrix

for pair in infect:

stack = [pair]

visited = [pair]

infect.remove(pair)

# DFS Algo and checking for neighbour infect's

while len(stack):

pos = stack.pop(-1)

# checking Right

if (pos[0], pos[1] + 1) in infect and (pos[0], pos[1] + 1) not in visited:

stack.append((pos[0], pos[1] + 1))

visited.append((pos[0], pos[1] + 1))

infect.remove((pos[0], pos[1] + 1))

# checking Left

if (pos[0], pos[1] - 1) in infect and (pos[0], pos[1] - 1) not in visited:

stack.append((pos[0], pos[1] - 1))

visited.append((pos[0], pos[1] - 1))

infect.remove((pos[0] - 1, pos[1] - 1))

# checking Up

if (pos[0] - 1, pos[1]) in infect and (pos[0] - 1, pos[1]) not in visited:

stack.append((pos[0] - 1, pos[1]))

visited.append((pos[0] - 1, pos[1]))

infect.remove((pos[0] - 1, pos[1]))

# checking Down

if (pos[0] + 1, pos[1]) in infect and (pos[0] + 1, pos[1]) not in visited:

stack.append((pos[0] + 1, pos[1]))

visited.append((pos[0] + 1, pos[1]))

infect.remove((pos[0] + 1, pos[1]))

# checking Upper Right

if (pos[0] - 1, pos[1] + 1) in infect and (pos[0] - 1, pos[1] + 1) not in visited:

stack.append((pos[0] - 1, pos[1] + 1))

visited.append((pos[0] - 1, pos[1] + 1))

infect.remove((pos[0] - 1, pos[1] + 1))

# checking Lower Right

if (pos[0] + 1, pos[1] + 1) in infect and (pos[0] + 1, pos[1] + 1) not in visited:

stack.append((pos[0] + 1, pos[1] + 1))

visited.append((pos[0] + 1, pos[1] + 1))

infect.remove((pos[0] + 1, pos[1] + 1))

# checking Upper Left

if (pos[0] - 1, pos[1] - 1) in infect and (pos[0] - 1, pos[1] - 1) not in visited:

stack.append((pos[0] - 1, pos[1] - 1))

visited.append((pos[0] - 1, pos[1] - 1))

infect.remove((pos[0] - 1, pos[1] - 1))

# checking Lower Left

if (pos[0] + 1, pos[1] - 1) in infect and (pos[0] + 1, pos[1] - 1) not in visited:

stack.append((pos[0] + 1, pos[1] - 1))

visited.append((pos[0] + 1, pos[1] - 1))

infect.remove((pos[0] + 1, pos[1] - 1))

counter.append(len(visited))

print(file\_name[:13], 'Output:')

print(max(counter), '\n') # Print max value of infected area

# Task 02

def apocalypse(file\_name):

# file input

with open(file\_name, 'r') as file:

row = file.readline()

col = file.readline()

matrix = [[ele for ele in line.split(' ')] for line in file]

# removing line break(\n)

for index in range(len(matrix)):

matrix[index][-1] = matrix[index][-1].replace('\n', '')

# creating a tuple list of the positions of aliens

alien = []

for row in range(len(matrix)):

for col in range(len(matrix[row])):

if 'A' in matrix[row][col]:

alien.append((row, col))

# creating a tuple list of the positions of humans

human = []

for row in range(len(matrix)):

for col in range(len(matrix[row])):

if 'H' in matrix[row][col]:

human.append((row, col))

visited = []

Flag = False

level = [] # to keep track of the time to kill

# Applying BFS on every pair o alien touple

for pair in alien:

queue = [pair]

count = 0

# BFS and checking right, left, up, down for neighbours

while len(queue):

pos = queue.pop(0)

# checking Right

if (pos[0], pos[1] + 1) in human and (pos[0], pos[1] + 1) not in visited:

queue.append((pos[0], pos[1] + 1))

visited.append((pos[0], pos[1] + 1))

Flag = True

# checking Left

if (pos[0], pos[1] - 1) in human and (pos[0], pos[1] - 1) not in visited:

queue.append((pos[0], pos[1] - 1))

visited.append((pos[0], pos[1] - 1))

Flag = True

# checking Up

if (pos[0] - 1, pos[1]) in human and (pos[0] - 1, pos[1]) not in visited:

queue.append((pos[0] - 1, pos[1]))

visited.append((pos[0] - 1, pos[1]))

Flag = True

# checking Down

if (pos[0] + 1, pos[1]) in human and (pos[0] + 1, pos[1]) not in visited:

queue.append((pos[0] + 1, pos[1]))

visited.append((pos[0] + 1, pos[1]))

Flag = True

# checker to count only when the queue is pushed when neighbour found.

if (Flag):

count += 1

Flag = False

level.append(count)

print(file\_name[:13], 'Output:')

print(f"Time: {max(level)} minutes")

if (len(visited) == len(human)):

print("No one survived",'\n')

else:

print(len(human) - len(visited), "survived",'\n')

covid\_tracer('Task01\_Input1.txt')

covid\_tracer('Task01\_Input2.txt')

# Must change Input file name according to your

apocalypse("Task02\_Input1.txt")

apocalypse("Task02\_Input2.txt")