#Task\_1\_of\_19101444

print("Output of Task 1:")

region\_matrix = []

with open('input.txt') as text:

    text\_lines = text.readlines()

    #print(text\_lines)

for text\_line in text\_lines:

    region\_matrix.append(text\_line.split())

#region\_matrix

for column in range(len(region\_matrix)):

    for row in range(len(region\_matrix[column])):

        if region\_matrix[column][row] =='Y':

            region\_matrix[column][row] = 1

        else:

            region\_matrix[column][row] = 0

#row\_size = len(region\_matrix)

#column\_size = len(region\_matrix[0])

def track(region):

    result = 0

    for c in range(len(region)):

        for r in range(len(region[c])):

            if region[c][r] == 1:

               counter = DFS(region, c, r)

               result = max(result, counter)

    return result

def DFS(region, c, r):

    if any([c<0, r<0, c>=len(region), r>=len(region[0])]):

        return 0

    if region[c][r] == 0:

        return 0

    count = 1

    region[c][r] = 0

    for column in range( c-1,c+2):

        for row in range( r-1, r+2):

            if any( [column!=c, row!=r]):

                count = count + DFS( region, column, row)

    return count

maximum\_region = track(region\_matrix)

print("Maximum region is:", maximum\_region)

"""

Here comes Task 2

"""

#TASK\_2\_of\_19101444

print("Output of Task 2:")

region\_matrix = []

with open('Question2 input2.txt') as text:

    text\_lines\_1 = text.readlines()[0:2]

    print(text\_lines\_1)

with open('Question2 input2.txt') as text:

    text\_lines\_2 = text.readlines()[2:]

    print(text\_lines\_2)

for text\_line in text\_lines\_2:

    region\_matrix.append(text\_line.split())

#region\_matrix

time=0

for column in range(len(region\_matrix)):

    for row in range(len(region\_matrix[column])):

        if region\_matrix[column][row] =='A':

            region\_matrix[column][row] = 1

        elif region\_matrix[column][row] =='H':

            region\_matrix[column][row] = 0

        else:

            region\_matrix[column][row] = -1

        #row\_size = len(region\_matrix)

        #column\_size = len(region\_matrix[0])

def track(region):

    result = 0

    for c in range(len(region)):

        for r in range(len(region[c])):

            if region[c][r] == 0:

               counter = BFS(region, c, r)

               result = max(result, counter)

    return result

def BFS(region, c, r):

    global time

    if any([c<0, r<0, c>=len(region), r>=len(region[0])]):

        return 0

    if region[c][r] == 0:

        return 0

    count = 0

    region[c][r] = 0

    position = [(-1, 0), (0, 1), (1, 0), (0, -1)]

    for column, row in enumerate(position):

            if any( [column!=c, row!=r]):

                region[column][row]= 1

                if region[c][r] == 0:

                    count = count +1

                    time= time+1

            count = count + BFS( region, column, row)

    return count

survived = track(region\_matrix)

print("Time:", time, "minutes")

print(survived, "survived")