g = open('Output.txt','w')  
#Mahdi Uddin Ahmed  
#20301469  
#S4  
#--------------PROBLEM1----------------#  
  
# Empty List to hold the 2D Array  
List1 = []  
# Reading the I/O file  
with open('Problem1/input.txt', 'r') as f:  
 # line by line Iteration  
 for line in f:  
 a = line.split() # converting every line to a list  
 List1.append(a) # adding the lists to another list  
f.close()  
  
# Empty List to hold the 2D Array  
List2 = []  
# Reading the I/O file  
with open('Problem1/input1.2.txt', 'r') as f:  
 # line by line Iteration  
 for line in f:  
 a = line.split() # converting every line to a list  
 List2.append(a) # adding the lists to another list  
f.close()  
  
# Output check  
'''  
for i in List1:  
 print(i)  
'''  
  
  
# Main function to check max affected  
#DFS Start  
def affected(matrix):  
 *"""  
 a = matrix  
 return a  
 """* # adjacent cell co ordinates  
 xi = [-1, -1, -1, 0, 0, 0, 1, 1, 1]  
 yi = [-1, 0, 1, -1, 0, 1, -1, 0, 1]  
 r = len(matrix) # row size  
 c = len(matrix[0]) # column size  
   
 #Exploring the children  
 def Navigate(i, j, matrix, count):  
  
 # check of not Y  
 if matrix[i][j] != 'Y':  
 return count  
  
 # if Y mark visited  
 count += 1  
 matrix[i][j] = 0  
 # recursively check up down left right corner for Y to find the maximum area  
 for x, y in zip(xi, yi):  
 #fully explore the children  
 if 0 <= i + x < r and 0 <= y + j < c:  
 #FIFO  
 count = max(count, Navigate(i + x, y + j, matrix, count)) # recursive call  
 return count  
  
 max\_cells = 0  
 for i in range(r):  
 for j in range(c):  
 #Send parent node  
 max\_cells = max(max\_cells, Navigate(i, j, matrix, 0))  
 return max\_cells  
  
  
O1 = str(affected(List1))  
O2 = str(affected(List2))  
g.write('Task1:\n')  
g.write('Case1\n')  
g.write(O1)  
g.write('\n')  
g.write('Case2\n')  
g.write(O2)  
g.write('\n\n')  
  
  
#--------------PROBLEM2----------------#  
  
List1 = []  
# Reading the I/O file  
with open('Problem2/Question2input1.txt', 'r') as f:  
 # line by line Iteration  
 List2 = []  
 for line in f:  
 b = line.strip()  
 List2.append(b)  
 a = line.split() # converting every line to a list  
 List1.append(a) # adding the lists to another list  
  
f.close()  
List3 = []  
# Reading the I/O file  
with open('Problem2/Question2input2.txt', 'r') as f:  
 # line by line Iteration  
 List4 = []  
 for line in f:  
 b = line.strip()  
 List4.append(b)  
 a = line.split() # converting every line to a list  
 List3.append(a) # adding the lists to another list  
  
f.close()  
List1 = List1[2:]  
'''  
for i in List1:  
 print(i)  
'''  
List2 = List2[0:2]  
#print(List2)  
count = 0  
  
List3 = List3[2:]  
'''  
for i in List1:  
 print(i)  
'''  
List4 = List4[0:2]  
#print(List2)  
  
  
def attack(Range, matrix):  
 # up down left right co ordinates  
 global A, Cc  
 xi = [0, 1, 0, 0, -1]  
 yi = [1, 0, 0, -1, 0]  
 r = int(Range[0]) # row size  
 c = int(Range[1]) # column size  
  
 # Navigation  
 def dead(i, j, matrix, r, c, count):  
 Cc = 0  
 # If alien found  
 if matrix[i][j] == 'A':  
  
 # check all children of the node  
 for x, y in zip(xi, yi):  
 if 0 <= i + x < r and 0 <= y + j < c and matrix[i + x][j + y] == 'H':  
 # explore the children  
 matrix, Cc = dead(i + x, j + y, matrix, r, c, count)  
 matrix[i + x][j + y] = 'A'  
 # else finish the attack, go back to parent, explore another child and update the counter  
 count += 1  
  
 '''  
 if i<=r-1 and j+1<=c-1 and matrix[i][j+1]=='H':  
 matrix[i][j+1] = 'A'  
 if i+1<=r-1 and j<=c-1 and matrix [i+1][j]=='H':  
 matrix[i+1][j]='A'  
 if i-1>=0 and matrix[i-1][j]=='H':  
 matrix[i-1][j]='A'  
 if j-1>=0 and matrix[i][j-1]=='H':  
 matrix[i][j-1]='A'  
 '''  
  
 return matrix, count  
  
 # Start BFS  
 for i in range(r):  
 for j in range(c):  
 A, Cc = dead(i, j, matrix, r, c, count)  
 return A, Cc  
  
g.write('Task2:\n')  
g.write('Case1\n')  
H = 0  
# BFS call  
B, Cc = attack(List2, List1)  
for i in B:  
 for j in i:  
 if j == 'H':  
 H += 1  
if H==0:  
  
 g.write('No one Survived\n')  
else:  
 g.write(str(H))  
 g.write(' Survived\n')  
g.write('Time:')  
g.write(str(Cc))  
g.write(' minutes\n')  
  
g.write("Case2\n")  
H = 0  
# BFS call  
B, Cc = attack(List4, List3)  
for i in B:  
 for j in i:  
 if j == 'H':  
 H += 1  
if H==0:  
  
 g.write('No one Survived\n')  
else:  
 g.write(str(H))  
 g.write(' Survived\n')  
g.write('Time:')  
g.write(str(Cc))  
g.write(' minutes\n')