from collections import deque

ROW = int(input())

COL = int(input())

class Point:

def \_\_init\_\_(self,x: int, y: int):

self.x = x

self.y = y

class queueNode:

def \_\_init\_\_(self,pt: Point, dist: int):

self.pt = pt

self.dist = dist

def isValid(row: int, col: int):

return (row >= 0) and (row < ROW) and (col >= 0) and (col < COL)

rowNum = [-1, 0, 0, 1]

colNum = [0, -1, 1, 0]

def BFS(mat, src: Point, dest: Point):

if ((mat[src.x][src.y]=="S" or mat[src.x][src.y]=="S")==False or (mat[dest.x][dest.y]=="S" or mat[dest.x][dest.y]=="S")==False):

return -1

visited = [[False for i in range(COL)]

for j in range(ROW)]

visited[src.x][src.y] = True

q = deque()

s = queueNode(src,0)

q.append(s)

while q:

curr = q.popleft()

pt = curr.pt

if pt.x == dest.x and pt.y == dest.y:

return curr.dist

for i in range(4):

row = pt.x + rowNum[i]

col = pt.y + colNum[i]

if (isValid(row,col) and (mat[row][col]=="S") == True and not visited[row][col]):

visited[row][col] = True

Adjcell = queueNode(Point(row,col),curr.dist+1)

q.append(Adjcell)

return -1

'''

3

3

2 1

0 2

M M S

M M S

M S S

'''

def main():

l1=input().split()

l2=input().split()

source = Point(int(l1[0]),int(l1[1]))

dest = Point(int(l2[0]),int(l2[1]))

mat = []

for i in range(0,ROW):

addl=input().split()

mat.append(addl)

dist = BFS(mat,source,dest)

if dist!=-1:

print("Shortest Path is",dist)

else:

print("Shortest Path doesn't exist")

main()