```
'''

        Time complexity: O(N*M)
        Space complexity: O(N*M)
        where N and M are the matrix parameters


'''
from sys import stdin
dir = [[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 1], [1, -1], [1, 0], [1, 1]]

pattern = ['C', 'O', 'D', 'I', 'N', 'G', 'N', 'I', 'N', 'J', 'A']

def validPo(x, y, n, m) :

        return (x >= 0 and x < n and y >= 0 and y < m)

def DFS(arr, used, x, y, index, n, m) :

    if (index == 11) :

        return 1

    used[x][y] = True
    found = 0

    for i in range(8) :

        newx = x + dir[i][0]
        newy = y + dir[i][1]

        if(validPo(newx, newy, n, m) and arr[newx][newy] == pattern[index] and used[newx][newy] == False) :

            found = found | DFS(arr, used, newx, newy, index + 1, n, m)

    used[x][y] = False

    return found


def solve(arr, n, m) :

    found=0
    used = [[False for i in range(m)] for j in range(n)]

    for i in range(n) :

        for j in range(m) :
            if (arr[i][j] == 'C') :

                found = DFS(arr, used, i, j, 1, n, m)

                if (found != 0) :

                    break

        if (found != 0) :

            break

    return found



def takeInput():
    #To take fast I/O
    n,m=list(map(int,stdin.readline().strip().split( )))
    arr = [stdin.readline().strip() for i in range(n)]
    return arr,n,m



# Main
arr,n,m=takeInput()
print(solve(arr,n,m))
```