#Luke is daydreaming in Math class. He has a sheet of graph paper with rows and columns, and he imagines that there is an army base in each cell for a total of bases. He wants to drop supplies at strategic points on the sheet, marking each drop point with a red dot. If a base contains at least one package inside or on top of its border fence, then it’s considered to be supplied. For example:

#image

#Given and , what’s the minimum number of packages that Luke must drop to supply all of his bases?

#Example

#Packages can be dropped at the corner between cells (0, 0), (0, 1), (1, 0) and (1, 1) to supply bases. Another package can be dropped at a border between (0, 2) and (1, 2). This supplies all bases using packages.

#Function Description

#Complete the gameWithCells function in the editor below.

#gameWithCells has the following parameters:

#int n: the number of rows in the game

#int m: the number of columns in the game

#Returns

#int: the minimum number of packages required

#Input Format

#Two space-separated integers describing the respective values of and .

#Constraints

#Sample Input 0

#2 2

#Sample Output 0

#1

#Explanation 0

#Luke has four bases in a grid. If he drops a single package where the walls of all four bases intersect, then those four cells can access the package:

#image

#Because he managed to supply all four bases with a single supply drop, we print as our answer.

#Python-Army Game

Import math

Import os

Import random

Import re

Import sys

def gameWithCells(n,m):

cells = math.ceil(n/2) \* math.ceil(m/2)

Return cells

If \_\_name\_\_ == ‘\_\_main\_\_’:

Fptr = open(os.environ[‘OUTPUT\_PATH’], ‘w’)

First\_multiple\_input = input().rstrip().split()

N = int(first\_multiple\_input[0])

M = int(first\_multiple\_input[1])

Result = gameWithCells(n, m)

Fptr.write(str(result) + ‘\n’)

Fptr.close()