Imagine a white rectangular grid of n rows andm columns divided into two parts by a diagonal line running from the upper left to the lower right corner. Now let's paint the grid in two colors according to the following rules:

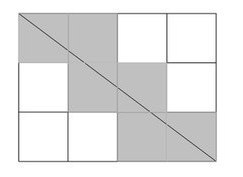
* A cell is painted black if it has at least one point in common with the diagonal;
* Otherwise, a cell is painted white.

Count the number of cells painted black.

**Example**

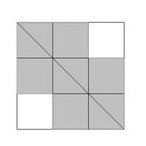
* For n = 3 and m = 4, the output should be  
  countBlackCells(n, m) = 6.

There are 6 cells that have at least one common point with the diagonal and therefore are painted black.



* For n = 3 and m = 3, the output should be  
  countBlackCells(n, m) = 7.

7 cells have at least one common point with the diagonal and are painted black.



**Input/Output**

* **[time limit] 500ms (cpp)**
* **[input] integer n**

The number of rows.

*Constraints:*  
1 ≤ n ≤ 105.

* **[input] integer m**

The number of columns.

*Constraints:*  
1 ≤ m ≤ 105.

* **[output] integer**

The number of black cells.

<https://codefights.com/arcade/code-arcade/loop-tunnel/RcK4vupi8sFhakjnh>

<http://pastebin.com/duuG8eGi>

1. int countBlackCells(int n, int m) {
2. // Switch so that n is always smaller then m
3. if ( n > m ) {
4. int buffer = n;
5. n = m;
6. m = buffer;
7. }
9. // Count base
10. int counter = m + n - 1;
12. // Add cross point cells
13. if ( m%n == 0 ) {
14. int gap = m/n;
15. counter += (m/gap)-1;
16. } else {
17. // Lowest commun multiple - I ended up not using lcm, but m/gcf
18. int lcm = m/GCF(m,n);
19. printf("%d**\n**", lcm);
20. if ( lcm != 1 ) {
21. counter += (m/lcm)-1;
22. }
23. }
25. return counter;
26. }
28. int GCF(int a, int b) {
29. if (b == 0) return a;
30. else return (GCF (b, a % b));
31. }