Name: Mohammad Ashaf khan
UID: 23BCS11123.

Q. Given three integer n, a and b; return nth magical no. since the ans may be vary large no. so return mod $10^9 + 7$.

A magical no — if no. is divide by a or b.

$n = 1$, $a = 2$, $b = 3$

output = 2.

No.	divisible by:	Magical
2	div by 2	✓
3	3	✓
4	2	✓
5	x	
6	2 and 3	✓
8		✓

Ans. magical no. (div by a or b)

→ multiple of a
→ multiple of b] but count only once

$$\text{Count}(x) = \left(\frac{x}{a} + \frac{x}{b} - \frac{x}{\text{lcm}(a,b)} \right)$$

Approach:

1. compute lcm (a, b)
2. The answer lies between ~~min(a,b)~~ 1 and n * min(a, b)
3. Use binary search to find nth magical no.
4. Return result % $(10^9 + 7)$

$n = 5$ $a = 2$ $b = 3$

2
3
4
5
6
7
8
12
9
18
24
30

Number	Divisible by2	Div by3	Magical
1	x	x	x
2	✓	x	✓
3	x	✓	✓
4	✓	x	✓
5	x	x	x
6	✓	✓	✓
7	x	x	x
8	✓	x	✓

for $n = 5$ output = 8

Brute force code

__/__/____

```
#include <bits/stdc++.h>
using namespace std;

int nthMagical (int n, int a, int b) {
    int count = 0;
    int num = 1;
    while (true) {
        if (num % a == 0  ||  num % b == 0) {
            count++;
            if (count == n) {
                return num;
            }
        }
        num++;
    }
    return -1;
}
```

optimal code in c++

// count (x) = $\underbrace{\frac{x}{a} + \frac{x}{b}}_{\text{Inclusion}} - \underbrace{\left(\frac{x}{lcm(a,b)}\right)}$

Exclude to prevent counting same no. twice
as using count to return
nth magical.

```
#includ <bits/std
using namespace std;
long long gcd ( long long a, long long b ) {
    while (b != 0) {
        long long t = a % b;
        a = b;
        b = t;
    }
    return a;
}
long long lcm (long long a, long long b) {
    return (a/gcd(a,b)) * b;
}

int main() {
    int n, a, b;
    cin >> n >> a >> b;
const int MOD= 1e9 + 7;
long long low = 1;
~~long long high = (long long) n * min(a, b);~~
long long L = lcm(a, b);
while ( low < high) {
    long long mid = low + (high - low) /2 ;
     long long count = mid /a + mid/b - mid/L ;
      if (count < n)
         low = mid+1 ;
      else
          high = mid;
  }
  cout << low % MOD ;
  return 0;
}
```