Name - Tanveet Kaur
UID - 23BCS11761
Section - KRG 3A.

Q Given three integer n, a and b, return n^{th} magical no. Since the ans may be very large return

A magical no. - if no. is divisible by either a or b.

n=1, a=2, b=3.
Out = 2

⇒ Approach:

1. Creating a modulo with value 10^9+7
2. Calculating lcm of a and b with the help of GCD relation

$$lcm = \frac{(a \times b)}{gcd(a,b)}$$

3. Performing binary search
   (start) s = min (a, b)
   (end) e = n x min (a, b)
   $mid = \frac{(s+e)}{2}$
4. Counting magical number ≤ mid.
   $count = \frac{mid}{a} + \frac{mid}{b} - \frac{mid}{lcm}$
   if count < n then s = mid+1
   else ⇒ e = mid.
5. return ans % MOD.

⇒ Code:

```
long long gcd (long long a, long b) {
    if (b == 0) return 0;
```

```
    return gcd(b, a%b);
}
long long lcm(long long a, long long b){
    return (a/(gcd(a,b)))*b;
}

int nthMagicalNumber(int n, int a, int b){
    int mod = 1000000007;

    long long low = min(a,b);
    long long high = 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/c;

        if(count < n)
            low = mid+1;
        else
            high = mid;
    }
    return low % MOD;
}

int main(){
    cout << nthMagicalNumber(1,2,3);
}.
```

OUTPUT = 2.