Name => Asmit Prabhakar
Uid => 23BCS10063
Section => KRG 2B.

_/_/__

(Q) Given three integers n, a, b, return n^{th} magical no. Ans can be very large return 10^9+7 %.
magical number → if no is either divisible by a or b.

Solution :-

Test case:
n = 1, a = 2, b = 3.
output := 2

Algorithm :-

i) Brute force :-

```
int i = min(a,b);
     int ans = 0
   while(n){
        count <
    if(i % a == 0 || i % b == 0){
          ans ++;
       }
         ~~n--;~~ if(count==n)
         i~~++~~;        return i % (10^9+7);
       }  ~~return ans % (10^9+7);~~
            i++;
         }
```

ii) Optimal approach :-
Algorithm :-
i) Here we can use Binary search algorithm to find magical number
ii) Now, compute low = min (a,b) and high = n * min(a,b)
iii) Find $LCM = \frac{a * b}{gcd(a,b)}$

iv) while (low < high){.

$$mid = low + \frac{high-low}{2};$$

iv) For each mid value
we will count magic numbers

$$\therefore magiccount = \left(\frac{mid}{a} + \frac{mid}{b} - \frac{mid}{lcm}\right);$$

```
vi) if (magiccount >= n){
            move to left.
        }
    else {
            move to right;
```

vi) Return magic number low % 10^9+7.

code :-

```
int findmagic(int a, int b, int c) {
        int low = mid(a,b);
        int high = n * min(a,b);
        int lcm = (a*b)/gcd(a,b);
        while(low ≤ high){
            mid = low + (high-low)/2;
            int count = (mid/a) + (mid/b) - (mid/lcm);
            if (count ≥ n){
                high = mid;
            } else
                low = mid+1
        }
        return low % (10^9+7);
}
```

Input :-
1, 2, 3

output :-
2.

Time complexity
$= O(\log min(a,b))$