Name: Amol Saurav
UID: 23BCS10067
Date: 30/1/2026

KRG-2B

Q) Given three integers n, a and b return n^{th} magical no. Ans may be very large return $10^9 + 7$ %

A number is a magical number if it is either divisible by a or b.

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

Brute force:

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

Optimal:

Algorithm:

(i) we will use Binary search to find the magic element
(ii) we will use low = min(a, b) and high = n * min(a, b)
(iii) find lcm(a, b) = $\frac{a * b}{gcd(a, b)}$
(iv) while low < high repeat the following steps
mid = low + $\frac{high - low}{2}$
(v) for each mid value
we will count magic numbers.
magicCount = $\frac{mid}{a} + \frac{mid}{b} - \frac{mid}{lcm}$

if (magiccount >= n)
go to left part
else
go to right part

(v) Return the magic number i.e low % 10^9+7

Code :-

```
int magicnumber(int a, int b, int n) {
    int low = min(a, b);
    int high = n* min(a, b);
    int lcm = (a*b) / gcd(a, b);
    while( low <= high) {
        int 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);
}

int gcd(int a, b) {
    if (b == 0) return a;
    return (b, a % b);
}
```

OUTPUT :-

a = 2, b = 3, n = 1

↳ Prints => 2

Time complexity

O(log(min(a, b)))