Name = Khanika
Khu/0652
UID = 23BCg-3A
Section -

classmate
Date ______
Page ______

Qu:- Given three integers n, a, and b return nth magical no since the ans may be very large - return mod(10^9+7).
A magical no - if it is divisible by either a or b

$n=1, a=2, b=3 \rightarrow$ Out :- 2

Brute - Force
Check no. one by one - and count magical no.

num % a OR num % b → count++

[TC = O(n x min(a,b))]

Optimal Approach :-

:→ Using inclusion - exclusion principle :-

$$\boxed{count(x) = x/a + x/b - x/lcm(a,b)}$$

:→ And we have to use Binary Search
Because - as x increases → count(x) increases monotonically.
and we want smallest x.
Search :- low :- min(a,b)
high = n* min(a,b)

:→ Algorithm :-
1) compute lcm :-
lcm(a,b) = (a/gcd(a,b))* b

2) Binary search :-
while (low<=high):
mid = (low+high)/2;
if (count(mid)>=n):
ans = mid;

high = mid-1
else
low = mid+1

Return ans % (1e9+7)

→ Code:-

```
static const in MOD = 1e9 + 7;
long long gcd( long long a , long long b){
  return b==0 ? a : gcd(b, a%b); }
int func( int n, int a, int b) {
  long long lcm = ( a/gcd(a,b))*b;

  long long low = min(a,b);
  long long high = n* min(a,b);
  long long ans=0;
  while ( low <= high){
  long long mid = low + (high - low) /2;
  long long count = mid/a + mid/b - mid/lcm
  if( (count >= n){
        ans = mid;
        high = mid-1; } else{
           low = mid+1; }}
     return ans % mod; }};
```

T.C $\boxed{O(\log(n \times \min(a,b))) \approx O(\log n)}$