```
Public static void main (String [] args) {
  Scaner sc= new Scanner (System.in)
   long n = nextLong();
    long a = nextLong();
     long b = next long();
     long magical = mag (n, a, b);
     System.out.println (" n^th magical no.: " + magical);
  }
}
```

Algorithm (optimal):

1) get n, a, b, initialize L, R, i, create array ar of size n-1.

2) L = min (a,b) , R = LCM (a,b)

3) find no. of magical no. k b/w range [L , R].

4) i = n/k.

5) $L += LCM(a,b)*i$, $R += LCM(a,b)*i$, find $t = k + (k*i-1)+1$;

6) You can find n^{th} magical no. in range [L, R], L is the t^{th} magical no.

7) iterate through range till n^{th} magical no. M found

```
    • a1 = L; , b1 = L + (b-a);
    • If (a<b)  ar[t] = a1, a1+= a1;
      else if (a>b)  ar[t] = b1, b1 += b1;
      else  ar[t] = a1, a1+= a1, b1 += b1
    • Iterate through above If, else if, else sequence till
```

n^{th} magical no. M found

8) return ar[$\overset{n-1}{\cancel{t}}$]; // n^{th} magical no.

9) End.

Name: Tanay Manish Nehari
UID: 23BCS13761
Class: 2B

Date |
Page |

CC.

Q) Given 3 integers n, a, b
return n^th magical no. Ans may be very large when 10^9 +7.
Magical no. - If no. is divisible by either 2 or 3
Eg: n = 1 , a = 2 , b = 3
output = 2

Sol) Algorithm:

1) Create an array ar of size n-1. int t = 0. take a and b as input from user.
2) Find smaller no. between a and b, initialize ~~i~~ i as equal to it.
3) Start iterating using loop from i under following condition till t = n-1:
   - If (i%a==0 || i%b==0) ar[t] = i;
   - ++t;
4) Return ar[t], as it is n^th magical no.
5) End

Code:

```
import java.util.*;
Public class main {
  Public static long mag(long n, long a, long b){
     long size = n-1;
     long t = 0;
     long[] ar= new ar[size];
     long i = min(a,b);
     while (t<n){
         If(i%a==0 || i%b==0){
             ar[t] = i;
              ++t;
         }
         ++i;
     }
     return ar[t];
  }
```