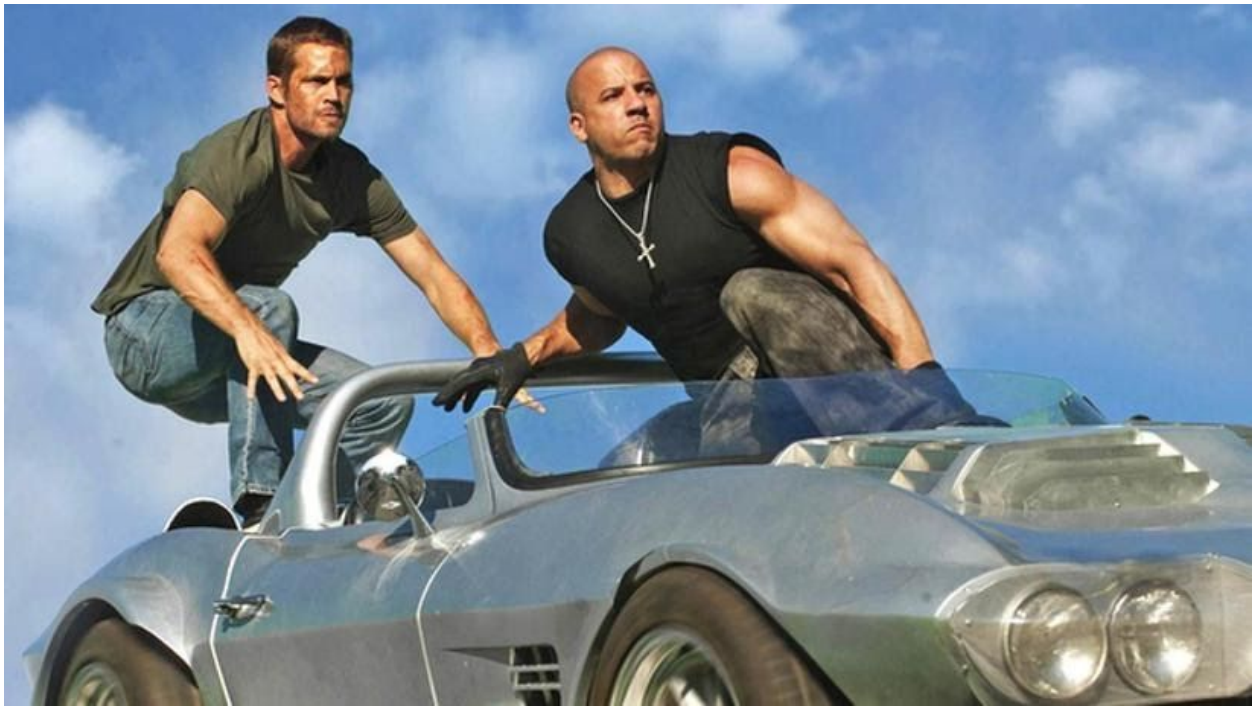
Help the Escape



Deckard Shaw has trapped Dominic Toreto and Brian O'Connor in a torture chamber which is getting smaller with each passing minute. In order to escape the chamber, they have to enter a code.

There are three numbers x,y and z flashing before them on a screen. Brian figures out they have to enter the n-th number which is divisible by either x,y or z as the code and set themselves free.

Help Brian and Dom escape, before it's too late.

Input Format

- The first line of input contains an integer n.

- The next 3 lines input integers x, y, z respectively.

Constraints

$1 \leq n, x, y, z \leq 10^9$
$1 \leq x \times y \times z \leq 10^{18}$
The results will be in the range $[1, 2 \times 10^9]$

Output Format

The ouput will be single integer.

Sample Input 0

```
100
8
11
15
```

Sample Output 0

```
390
```

Expected Solution : (Java8)

```
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;
import java.util.StringTokenizer;

public class HelpTheEscape {
```

```
static class Reader
{
    final private int BUFFER_SIZE = 1 << 16;
    private DataInputStream din;
    private byte[] buffer;
    private int bufferPointer, bytesRead;

    public Reader()
    {
        din = new DataInputStream(System.in);
        buffer = new byte[BUFFER_SIZE];
        bufferPointer = bytesRead = 0;
    }

    public Reader(String file_name) throws IOException
    {
        din = new DataInputStream(new FileInputStream(file_name));
        buffer = new byte[BUFFER_SIZE];
        bufferPointer = bytesRead = 0;
    }

    public String readLine() throws IOException
    {
        byte[] buf = new byte[64]; // line length
        int cnt = 0, c;
        while ((c = read()) != -1)
        {
            if (c == '\n')
                break;
            buf[cnt++] = (byte) c;
        }
        return new String(buf, 0, cnt);
    }

    public int nextInt() throws IOException
    {
        int ret = 0;
        byte c = read();
        while (c <= ' ')
            c = read();
        boolean neg = (c == '-');
        if (neg)
```

```
        c = read();
    do
    {
        ret = ret * 10 + c - '0';
    }  while ((c = read()) >= '0' && c <= '9');

    if (neg)
        return -ret;
    return ret;
}

public long nextLong() throws IOException
{
    long ret = 0;
    byte c = read();
    while (c <= ' ')
        c = read();
    boolean neg = (c == '-');
    if (neg)
        c = read();
    do {
        ret = ret * 10 + c - '0';
    }
    while ((c = read()) >= '0' && c <= '9');
    if (neg)
        return -ret;
    return ret;
}

public double nextDouble() throws IOException
{
    double ret = 0, div = 1;
    byte c = read();
    while (c <= ' ')
        c = read();
    boolean neg = (c == '-');
    if (neg)
        c = read();

    do {
        ret = ret * 10 + c - '0';
    }
    while ((c = read()) >= '0' && c <= '9');
```

```
        if (c == '.')
        {
            while ((c = read()) >= '0' && c <= '9')
            {
                ret += (c - '0') / (div *= 10);
            }
        }

        if (neg)
            return -ret;
        return ret;
    }

    private void fillBuffer() throws IOException
    {
        bytesRead = din.read(buffer, bufferPointer = 0, BUFFER_SIZE);
        if (bytesRead == -1)
            buffer[0] = -1;
    }

    private byte read() throws IOException
    {
        if (bufferPointer == bytesRead)
            fillBuffer();
        return buffer[bufferPointer++];
    }

    public void close() throws IOException
    {
        if (din == null)
            return;
        din.close();
    }
}

public static void main(String[] args) throws IOException {
    Reader rd = new Reader();
    int n = rd.nextInt(), A = rd.nextInt(), B = rd.nextInt(), C = rd.nextInt();
    System.out.println(nthNumber(n, A, B, C));
}

public static int nthNumber(int n, int A, int B, int C) {
```

```
        if(A==1||B==1||C==1)return n;
        long res=0;
        long a=A;
        long b=B;
        long c=C;
        long ab=a*b/gcd(a,b);
        long ac=a*c/gcd(a,c);
        long bc=b*c/gcd(b,c);
        long abc=a*bc/gcd(a,bc);
        long left=0;long right=2000000000;
        while(left<=right){
            long mid=left+(right-left)/2;
            long cnt=mid/a+mid/b+mid/c-mid/ab-mid/ac-mid/bc+mid/abc;
            if(cnt==n){
                res=mid;
                right=mid-1;
            }
            else if(cnt>n){
                right=mid-1;
            }else{
                left=mid+1;
            }
        }
        return (int)res;
    }

    public static long gcd( long a, long b) {
        if( a < b) return gcd(b,a);
        if( b == 0 ) return a;
        return gcd(b,a%b);
    }
}
```