

# Números primos y divisibilidad: LCM

```
from math import sqrt

def criba(n):
    sieve = [True] * (n + 1)
    sieve[0], sieve[1] = False, False

    primes = []

    for i in range(2, int(sqrt(n)) + 1):
        if sieve[i]:
            primes.append(i)
            for j in range(i * i, n + 1, i):
                sieve[j] = False

    # Los que quedan con true despues de sqrt(n) son todos primos
    primes.extend([i for i in range(int(sqrt(n)) + 1, n + 1) if sieve[i]])

    return primes

def main():
    limit = int(1e6)

    # Criba de Eratóstenes
    numeros_primos = criba(limit)

    # Potencias de numeros primos.
    # Esto luego nos sirve para obtener las maximas potencias de los numeros primos,
    usadas para calcular el lcm.
    potencias = {}
    for prime in numeros_primos:
        potencia = prime
        while potencia < limit + 1:
            potencias[potencia] = prime
            potencia *= prime

    lcm = [0] * (limit + 1)
    potencias_de_dos = 0
    lcm_acumulado = 1

    for numero in range(1, limit + 1):
        # Si el numero no es una potencia maxima, significa que al descomponerlo sus primos
        tienen potencias menores o iguales a las maximas
        if numero in potencias:
```

```


if potencias[numero] == 5:
    # Eliminamos una potencia de 2 ya que se forma un 10:
    # Al hacer modulo 10 daria 0 y si dividimos por 10 (quitar una
    # potencia de 2 y una de 5), luego de que se realice el modulo, no da correcto.
    potencias_de_dos -= 1
elif potencias[numero] == 2:
    # Se necesita llevar la contabilidad de potencias_de_dos,
    # ya que no da el resultado correcto al dividir por 2 luego
    # de que se haya hecho la multiplicacion por 2 y luego el modulo
    potencias_de_dos += 1
else:
    # Por propiedades del modulo => (((A*B)*C)...)*D) MOD 10 = ((((((A*B)MOD
10)*C)MOD 10)...)(MOD 10)*D) MOD 10
    lcm_acumulado *= potencias[numero]
    lcm_acumulado %= 10

lcm[numero] = (lcm_acumulado * pow(2, potencias_de_dos, 10)) % 10

results = []
while True:
    try:
        n = int(input())
        if n == 0:
            break
        results.append(str(lcm[n]))
    except EOFError:
        break
print("\n".join(results))

if __name__ == "__main__":
    main()

```

| Status   | Time   | Length | Lang        | Submitted           | Open                                | Share text  | RemoteRunId |
|----------|--------|--------|-------------|---------------------|-------------------------------------|--|-------------|
| Accepted | 1610ms | 2411   | PYTH3 3.5.1 | 2024-11-21 16:30:52 | <input checked="" type="checkbox"/> | <input type="checkbox"/>   | 29979904    |