First let us see the definition of **Euler's Totient Function.**
Euler's totient function denoted as **φ(n)**, is an arithmetic function that counts the positive integers less than or equal to **N** that are relatively prime to **N**.

For this question we need to find the sum of all **φ(n)** for all **i** between **1** to **N**. We see that the constraints are very high and we have multiple test cases, so we would prefer to pre-process all the values in one go so that we can answer each query in **O(1) time**.

How can we find **φ(N)** for a single **N**?
Properties of Euler's Totient Function

1. If **N** is prime then **φ(N) = N-1**, as all numbers less than **N** are relatively prime to **N**.
2. If **N** is prime and **k > 0**, then $\varphi(N^k) = N^k - N^{k-1}$.
3. If **x** and **y** are relatively prime, then **φ(x.y) = φ(x) x φ(y)**.

Now we will use these properties to find **φ(N)** when **N** is not prime.

$N = P_1^{k_1}.P_2^{k_2}...P_m^{k_m}$, where P_i are the prime factors of **N**.

$\varphi(N) = \varphi(P_1^{k_1}).\varphi(P_2^{k_2})...\varphi(P_m^{k_m})$

Let $X = P_1^{k_1}.P_2^{k_2}...P_{m-1}^{k_{m-1}}$

$\varphi(N) = \varphi(X).\varphi(P_m^{k_m})$

$\varphi(N) = (P_1^{k_1} - P_1^{k_1-1})(P_2^{k_2} - P_2^{k_2-1})....(P_m^{k_m} - P_m^{k_m-1})$

$\varphi(N) = \varphi(X)(P_m^{k_m}(1-(1/P_m)))$

Here we see that **X** will always be less than **N** thus if we use dynamic programming to solve for **φ(N)** we will always have the value of **φ(X)**. All that remains is to findP_m and $P_m^{k_m}$.

$P_m^{k_m} = N/X$ In order to find P_m we can simply modify theSieve of Eratosthenes to store the largest prime that divides any number.

```
for (int i=2; i <= MAX; i++)
    if (prime[i] == 0)
        for (int j = i + i; j < = MAX; j += i)
            prime[j] = i;
```

If **prime[N]** is equal to **0**, then the **N** is prime or else it represents the largest prime that divides **N**. When**prime[N]** is not equal to **0**, we can find **X** by diving **N** with**prime[N]** until **N** is divisible by **prime[N]**. Now since we have all the requisites to calculate **φ(N)** we can do it dynamically by iterating from **1** to **N**.

Now once we have the values of **φ(N)** for all possible **N**, we can simply pre-process the sum till the i^{th} position so that we can answer each query in **O(1)** after preprocessing of **O(N log N)**.