```cpp
#include "../bits/stdc++.h"

// sieve of eratosthenes
// primes[i] != 0 if i is prime
std::vector<int> sieve_of_eratosthenes(int n)
{
    std::vector<int> primes(n);
    for (int i = 2; i < n; ++i)
        primes[i] = i;
    for (int i = 2; i * i < n; ++i)
        if (primes[i])
            for (int j = i * i; j < n; j += i)
                primes[j] = 0;
    return primes;
}

// 素因数分解, overflow に注意
// verified: http://judge.u-aizu.ac.jp/onlinejudge/review.jsp?rid=3381567
std::map<int, int> prime_factorization(int n)
{
    std::map<int, int> ret;
    int ntmp = n;
    for (int i = 2; i * i <= n; i++)
    {
        if (ntmp % i)
            continue;
        while (ntmp % i == 0)
        {
            ret[i]++;
            ntmp /= i;
        }
    }
    if (ntmp != 1)
    {
        ret[ntmp]++;
    }
    return ret;
}

/**
 * 区間篩
 * [a, b) (abs(a-b)<=1e6 ぐらい) の素数表を得る
 * i is prime <=> ret[i-a] = true
 */
using ll = long long;
std::vector<bool> segment_sieve(ll a, ll b)
{
    // [2, sqrt(b)) の篩
    std::vector<bool> sm(sqrt(b) + 10, true);
    std::vector<bool> ret(b - a + 10, true);

    for (ll i = 2; i * i < b; i++)
    {
        if (sm[i])
        {
            for (ll j = 2; j * j < b; j += i)
                sm[j] = false;
            for (ll j = std::max(2LL, (a + i - 1) / i) * i; j < b; j += i)
                ret[j - a] = false;
        }
    }
    return ret;
}

```