```cpp
ll powll(ll n, ll p) {
  if (p == 0) return 1;
  else if (p == 1) return n;
  else {
    ll ans = powll(n, p / 2);
    ans = ans * ans;
    if (p % 2 == 1) ans = ans * n;
    return ans;
  }
}

// n = 1.5e7 -> 80 ms
vll list_prime_until(ll n) {
  vll ret;
  vector<bool> a(n + 1, true); // is_prime
  if (a.size() > 0) a[0] = false;
  if (a.size() > 1) a[1] = false;
  Loop(i, n + 1) {
    if (a[i]) {
      ret.push_back(i);
      ll k = (ll)i * i;
      while (k < n + 1) {
        a[int(k)] = false;
        k += i;
      }
    }
  }
  return ret;
}

// primes has to be generated by list_prime_until(>=sqrt(n))
vector<Pll> prime_factorize(ll n, const vll &primes) {
  vector<Pll> ret;
  Loop(i, primes.size()) {
    if (n == 1) break;
    while (n % primes[i] == 0) {
      if (ret.size() == 0 || ret.back().fst != primes[i]) {
        ret.push_back({ primes[i], 0 });
      }
      ret.back().snd++;
      n /= primes[i];
    }
  }
  if (n != 1) ret.push_back({ n, 1 });
  return ret;
}

vll divisors(const vector<Pll> factors) {
  queue<ll> que;
  que.push(1);
  Loop(i, factors.size()) {
    ll x = factors[i].fst, d = factors[i].snd;
    vll a(d + 1, 1); Loop1(j, d) a[j] = a[j - 1] * x;
    int m = int(que.size());
    Loop(j, m) {
      ll y = que.front(); que.pop();
      Loop(k, d + 1) que.push(y * a[k]);
    }
  }
  int m = int(que.size());
  vll ret(m);
  Loop(i, m) {
    ret[i] = que.front(); que.pop();
  }
  sort(ret.begin(), ret.end());
  return ret;
}
```