```cpp
/*
 * Multiplication by repeated addition, with fraction handling.
 *   by: Jim McCann, TCHOW llc
 *       ix@tchow.com
 *
 * They say it’s a grand challenge [1], but seems easy enough to me.
 * Just define multiplication in terms of addition!
 *
 * Compile: g++ -Wall -Werror -std=c++20 -o mul mul.cpp -pthread
 *
 * [1] "What, if anything, does multiplication even mean?",
 *      McCann, Jim. SIGBOVIK 2022.
 *
 */

#include <iostream>
#include <thread>
#include <chrono>
#include <random>
#include <list>

// This is a "constant-time" operation:
// it takes about ceil(b) seconds, even on an arbitrarily-fast processor.
double multiply(double a, double b) {
    //optimization:
    if (a == 0.0 || b == 0.0) return 0.0;

    std::atomic< double > sum = 0.0;
    std::list< std::jthread > threads;

    //NOTE: increase thread count for more accuracy
    for (uint32_t iter = 0; iter < 128; ++iter) {
        threads.emplace_back([&](std::stop_token stop){
            auto now = std::chrono::high_resolution_clock::now();
            std::random_device rd;
            std::mt19937 mt(rd());
            std::uniform_real_distribution<> uniform(0.0, 1.0);
            while (!stop.stop_requested()) {
                std::this_thread::sleep_until(now
                    + std::chrono::duration< double >(uniform(mt)));
                if (stop.stop_requested()) break;
                sum.fetch_add(a, std::memory_order_relaxed);
                now += std::chrono::seconds(1);
                std::this_thread::sleep_until(now);
            }
        });
    }

    std::this_thread::sleep_for(std::chrono::duration< double >(b)); //times b

    //optimization:
    for (auto &t : threads) t.request_stop();

    return std::scalbn(sum, -7);
}

int main(int argc, char **argv) {
    if (argc != 3) {
        std::cerr << "Usage:\n\t" << argv[0] << " <a> <b>\n"
            "Prints a * b. Supports fractions." << std::endl;
        return 1;
    }
    double a = atof(argv[1]);
    double b = atof(argv[2]);

    std::cout << multiply(a,b) << std::endl;

    return 0;
}
```