Token Access Program

```
#include <iostream>
#include <chrono>
#include <thread>
class TokenBucket {
    double tokens;
    double maxTokens;
    double refillRate; // tokens per second
    std::chrono::steady_clock::time_point lastRefill;

public:
    TokenBucket(double rate, double burst)
        : tokens(burst), maxTokens(burst), refillRate(rate),
          lastRefill(std::chrono::steady_clock::now()) {}

    void refill() {
        auto now = std::chrono::steady_clock::now();
        std::chrono::duration<double> elapsed = now - lastRefill;
        double addTokens = elapsed.count() * refillRate;
        if (addTokens > 0) {
            tokens = std::min(maxTokens, tokens + addTokens);
            lastRefill = now;
        }
    }
    bool consume(double reqTokens = 1) {
        refill();
```

```
        if (tokens >= reqTokens) {
            tokens -= reqTokens;
            return true;
        }
        return false;
    }
};
int main() {
    TokenBucket bucket(5, 10); // 5 tokens/sec, max burst 10

    for (int i = 0; i < 5; ++i) {
        if (bucket.consume()) {
            std::cout << "Request " << i << " allowed" << std::endl;
        } else {
            std::cout << "Request " << i << " denied" << std::endl;
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(200));
    }
}
```

Output

```
Request 0 allowed
Request 1 allowed
Request 2 allowed
Request 3 allowed
Request 4 allowed
Request 5 allowed
```