```cpp
#ifndef METNUM_TP2_COUNTER_H
#define METNUM_TP2_COUNTER_H

#include <string>
#include <list>
#include <map>
#include <chrono>
#include <fstream>

class Counter;
class Timer;

class Logger {
    friend class Counter;
    friend class Timer;
public:
    static Logger &getInstance() {
        static Logger instance;
        return instance;
    }

    void dump(std::string file) {
        std::fstream output(file, std::ios_base::out);

        for (auto &it : this->counters) {
            output << it.first << "\t\t\t";

            for (auto &lst : it.second) {
                output << lst << " ";
            }

            output << std::endl;
        }

        output.close();
    }
private:
    void set(std::string name, long long x) {
        try {
            std::list<long long> &temporal(this->counters.at(name));

            temporal.pop_front();
            temporal.push_front(x);
        } catch(...) {
            std::list<long long> empty;
            empty.push_back(x);
            counters.insert(std::pair<std::string, std::list<long long>>(name, empty));
        }
    }

    void reset(std::string name, long long x = 0) {
        try {
            std::list<long long> &temporal(this->counters.at(name));
            temporal.push_front(x);
        } catch(...) {
            std::list<long long> empty;
            empty.push_back(x);
            counters.insert(std::pair<std::string, std::list<long long>>(name, empty));
        }
    }

    Logger() {};
    Logger(Logger const&) = delete;
    void operator=(Logger const&)  = delete;

    std::map<std::string, std::list<long long>> counters;
};

class Counter {
public:
    Counter(std::string name, long long i = 0) : name(name), i(i) { }

    std::string inline getName() const {
        return this->name;
    }

    operator long long() const {
        return this->i;
```

```
79      }
80
81      Counter &operator+=(const Counter &m) {
82          this->i += m.i;
83          return *this;
84      }
85
86      Counter &operator-=(const Counter &m) {
87          this->i -= m.i;
88          return *this;
89      }
90
91      Counter &operator++() {
92          this->i++;
93          return *this;
94      }
95
96      Counter &operator--() {
97          this->i++;
98          return *this;
99      }
100
101     void set(long long x) {
102         i = x;
103     }
104
105     ~Counter() {
106         Logger::getInstance().reset(this->name, i);
107     }
108 private:
109     std::string name;
110     long long i;
111 };
112
113 class Timer {
114 public:
115     Timer(std::string name) : name(name), start(std::chrono::steady_clock::now()),
    end(std::chrono::steady_clock::now()), stopped(false) { }
116
117     std::string inline getName() const {
118         return this->name;
119     }
120
121     void reset(bool write = false) {
122         this->end = std::chrono::steady_clock::now();
123
124         if (write) {
125             Logger::getInstance().reset(this->name, std::chrono::duration_cast<std::chrono::microseconds>
    (this->end - this->start).count());
126         }
127
128         this->start = std::chrono::steady_clock::now();
129         this->stopped = false;
130     }
131
132     void stop() {
133         if (stopped) {
134             throw new std::runtime_error("Tried to stop an already stopped timer.");
135         }
136
137         this->stopped = true;
138         this->end = std::chrono::steady_clock::now();
139         Logger::getInstance().reset(this->name, std::chrono::duration_cast<std::chrono::microseconds>(this-
    >end - this->start).count());
140     }
141
142     ~Timer() {
143         if (!stopped) {
144             this->end = std::chrono::steady_clock::now();
145             Logger::getInstance().reset(this->name, std::chrono::duration_cast<std::chrono::microseconds>
    (this->end - this->start).count());
146         }
147     }
148 private:
149     std::string name;
150     std::chrono::steady_clock::time_point start;
151     std::chrono::steady_clock::time_point end;
152     bool stopped;
```

```
153 };
154
155 #endif //METNUM_TP2_COUNTER_H
```