National Research University Higher School of Economics
Faculty of Computer Science - Degree in Data Science and Business Analytics (DSBA)
Introduction to Programming

Homework (for bonus points) – Practice for programming test

Consider the file `earthquakes.csv` with data about earthquakes that occurred around the world during a day.

Part 1. Code two C++ structures `Time` and `Earthquake`. The structure `Time` represents a time of the day with hour, minutes, and seconds precision. You can use the templates below. For this homework:
- No encapsulation is required (e.g., no use of private attributes, getters/setters).
- Your solution (structures and main code) can be submitted in a single file.

```
struct Time
{
    int hour;
    int mins;
    int seconds;
    int toSeconds() const {
        // convert me to seconds...
    }
    bool operator<(const Time& t2) const
    {
        // am I "earlier" than t2 ?
    }
};
```

```
struct Earthquake
{
    double lat;
    double lon;
    double depth; // kms below the surface
    double magnitude; // in Richter scale
    std::string location;
};
```

1.1. Complete the method toSeconds(), which converts the hour, mins, seconds of a Time structure in a single integer as seconds. For example, if hour=23, mins=15, seconds=10, then toSeconds() return the integer 83710.

1.2. Complete the overload of < to compare which of two Time structures is earlier, e.g., 23:15:10 < 23:59:59.
For this, you could call the toSeconds() to compare Time structures.

1.3. Overload standard input (>>) for the structure Time.
If the user inserts the string 23:15:10 then the Time structure will have hour=23, mins=15, and seconds=10.

1.4. Overload the standard output (<<) for the structure Time.
If the Time structure has hour=23, mins=15, and seconds=10, then you must print 23:15:10.

1.5. Overload the standard output (<<) tor the structure Earthquake according to the example below:

```
lat=55.152 lon=160.503 depth=10 magnitude=4.5 location=74 km SE of Atlasovo-Russia
```

Part 2. Implement a method `fillMap(std::multimap<Time,Earthquake>& m)` that fills a container `std::multimap<Time,Earthquake> m`. That is, elements are key-value pairs, where keys are Time structures informing the time of an earthquake, and the values are Earthquake structures.

Part 3. Implement a method:

```
void queryEarthquakes(std::multimap<Time, Earthquake>& m, const Time& tstart, const Time& tend, std::string x)
```

- That prints in the console, and deletes from the multimap, all Earthquake structures that occurred in the interval [tstart, tend) and whose location attribute includes the substring x. If no Earthquakes meeting these conditions are found, print the string “No matches.”

Help: to iterate through the elements whose keys are in the interval [tstart, tend) you can use the functions upper_bound and lower_bound from maps.

- The arguments tstart, tend, and the string x are read by console. See an input/output example below.

INPUT:
```
10:00:00
23:59:59
Russia
```

OUTPUT:
```
lat=49.0958 lon=156.238 depth=35.12 magnitude=4.9 location=175 km S of Severo-Kuril’sk-Russia
lat=49.0136 lon=156.294 depth=10 magnitude=5.1 location=185 km S of Severo-Kuril’sk-Russia
lat=54.7653 lon=163.481 depth=10 magnitude=4.3 location=174 km SSE of Ust’-Kamchatsk Staryy-Russia
lat=54.9124 lon=162.529 depth=10 magnitude=4.4 location=146 km S of Ust’-Kamchatsk Staryy-Russia
```