Weather Stations

Introduction

The goal of this exercise is to load a file with weather station data and then print for each station the average temperature with standard deviation, minimum and maximum temperature. For the exercise you will be using C++ and CMake to build the program.

Project Structure

```
weather-stations/
├── CMakeLists.txt        # CMake configuration file
├── Makefile              # Make build configuration
├── src/
│   ├── main.cpp          # Main program file that reads and processes measurements
│   ├── cities.cpp        # Contains database of cities with their mean temperatures
│   ├── lz4.c             # The lz4 compression algorithm
│   ├── lz4.h             # The lz4 compression algorithm header
│   └── create-samples.cpp  # Program to generate sample measurement data
├── build/                # Build output directory (generated)
│   ├── cle-ws            # Main program executable
│   └── cle-samples       # Sample generator executable
```

Build

Run the following command to build the program:

```bash
make
```

Run the following command to run the program:

```bash
./build/cle-ws <input_file>
```

Input Data Format

The program reads a binary file containing compressed blocks of weather measurements. Each line of the uncompressed data block has the following format:

- Station name and temperature value separated by semicolon (;)
- Example: `Tokyo;15.4`

Input value ranges:

- Station name: UTF-8 string (1-100 bytes), containing neither semicolon (;) nor newline characters
- Temperature: double between -99.9 and 99.9 (inclusive), always with one decimal place
- Maximum of 10,000 unique station names
- Lines end with '\n' character on all platforms

Binary Format Specification

The file consists of a header followed by a sequence of data blocks. All integers are stored in Little-Endian format.

Structure Overview

Field	Description	Type
`NumBlocks`	Total number of blocks to follow	Int32
`Blocks`	An array of data blocks	Block[...]

Block Layout

Each block is structured as follows:

- **Compressed Size**: `Int32`
- **Original Size**: `Int32`
- **Data**: `Byte[Compressed Size]`

Requirements

1) Modify the `main.cpp` file to:
    - Read the input file (provided as command line argument or default to `measurements-1.cle`)
    - For each station, calculate:
        - Average temperature
        - Temperature Standard deviation
        - Minimum temperature
        - Maximum temperature
    - Print results sorted alphabetically by standard deviation

2) Expected JSON output format for each station:

```json
{
  "cities": [
    {
      "name": "London",
      "avg": 10.8,
      "std": 0.6,
      "min": 10.2,
      "max": 11.3
    },
    {
      "name": "Tokyo",
      "avg": 15.8,
      "std": 0.4,
      "min": 15.4,
      "max": 16.1
    }
  ]
}
```

Memorize

A proper formatting of the JSON output is not required but a proper JSON object is required.

Sample Data Generation

A program to generate sample data is provided (`cle-samples`). Usage:

```bash
./build/cle-samples NUMBER_OF_BLOCKS
```

This will create a file named `measurements-NUMBER_OF_BLOCKS.cle` containing random temperature measurements based on real city data. The data is compressed into blocks to reduce the amount of memory required to store it. Each block has an original (uncompressed) size of approximately 512 MiB.

As the number of blocks increases, the program will take longer to generate the data, and the output file will become larger.

The ultimate goal is to process one billion samples or more as quickly as possible, which corresponds to approximately 90 blocks of data. However, you should start by processing at least one block of samples, then gradually increase the number of blocks until the program takes too long to run.

Tip

- Consider using appropriate data structures to store and process the measurements
- Remember to handle file reading errors appropriately
- Pay attention to floating-point number formatting in the output
- For ordering the output, consider using a map or a sorted container