Lab 6 – Pointers, Makefile

Assume that a sensor network is used to monitor the temperature of all classrooms of Group T. All sensors report their measurements to the sensor gateway, an embedded device that provides local storage and a first processing of the data. The sensor gateway is also connected to the internet, but that connection is only relevant for the exercises of the following labs. The sensors can measure the temperature in steps of 0.1°C and send their measurements in packets of 4 bytes. Each packet has the following packet structure:

- The first 12 bits (bit 0 to bit 11) contain the sensor ID; the sensor network supports at most 4096 sensors but not all IDs are needed to monitor Group T;
- The next 6 bits (bit 12 to bit 17) are a 4 bit packet sequence number and 2 bit flags but these bits will not be used in this exercise and can be ignored;
- The next 13 bits (bit 18 to bit 30) encode the temperature value; bit 18 represents the sign bit (0 is positive sign, 1 is negative sign); the next 12 bits represent 4 octal digits that are the octal representation of the absolute value of a temperature multiplied by 10; For example:
    - \o147 equals 103 which represents 10.3°C
    - \o777 equals 511 which represents 51.1°C
- bit 31, the last bit, is the even-parity check bit

(Hint for the packet data structure: struct/union with bit fields.)

The sensor gateway captures incoming packets of sensor nodes and immediately attaches to every packet a time-stamp. It also performs an even-parity check and packets that fail this check will be discarded.

The sensor gateway maintains for every sensor a separate pointer list. You can use an array ranging over all sensor IDs to maintain all these pointer lists. For each incoming packet, the sensor ID is read and the sensor data with time-stamp is stored in the corresponding pointer list.

From time to time (e.g. this could be daily or weekly, depending on the measurement frequency of the sensors) the sensor gateway will run a 'data compression' tool that substitutes every 3 successive sensor measurements with only 1 measurement containing the average of all 3. Since this is repeated over and over again in time, it prevents that the memory used per sensor is growing out of proportion. Of course, older measurements will become coarser and coarser, and information will be lost.

Your task is to implement the sensor gateway software using the shared library implementation of a pointer list developed in the previous lab. You will test and run the software on a Linux desktop PC and not on a real embedded device.

You should also implement a small program that simulates the input of the sensor gateway as we will not interface with a real sensor network. However, using the pseudo-random generator and the sleep function (read man pages of sleep and usleep), it should not be a real problem to implement a program generating sensor data packets at a pre-defined frequency.
Linux command-line redirection can be used to connect both programs. E.g., type "**sensor_simulator | sensor_gateway**" on the command-line to redirect the sensor data packets of the sensor simulator to the input of the sensor gateway.