CST8234 C Programming

Lab Exercise #4

Purpose To gain experience with:

- using structures (e.g. `struct`, `typedef`) and standard library functions,
- creating solutions consisting of multiple functions and files,
- managing multi-file compilation using `make` utility and `makefile` in standard format.

Activity

- You are provided with the log file from an IoT device with multiple weather sensors.
  - It contains 3 different types of messages in CSV format (i.e. Comma Separated Values).
- **Task #1** – read the file line by line from stdin using redirection
  - e.g. mypgm < sensors_output.txt
- **Task #2** – create a structure to represent each type of message
  - The 3 message types in given below:

Sensor	Binary types	Legend
Temp	`time_t,1,double`	Timestamp, Type, Celsius
Wind speed & direction	`time_t,2,double,int`	Timestamp, Type, KmPerH, Degrees
Wind gusts	`time_t,3,double,double`	Timestamp, Type, Prev, Current

- **Task #2** – identify each message type read,
  - using the legend/guide, convert each field of the message to its corresponding binary types
  - create an instance of the correct structure and store the converted message in it.
- **Task #3** – group the structures together
  - place each instance of a structure in an array. There should be 1 array for each type of message/structure.

- **Task #4** – print and summarize the data
  - print each array in tabular format
  - convert the time stamp to the following format: yyyy/mm/dd hh:mm:ss
  - for the *Wind gust* messages, add a column that indicates whether the wind is increasing, decreasing or unchanged.
- **Hints**:
  - Use scanf() or sscanf() or use standard functions from string.h to process strings (e.g. strtok()) and stdlib.h (e.g. atoi())
  - You may assume the input file is already sorted by the time stamp field

Coding requirements:

- **There is no end-user input;** there should only be output
  - use printf() or sprintf() for all formatted output
- Your solution must consist of:
  - a .c file for each of message types and its corresponding .h file
  - a .c file for main()
  - a makefile in standard make format
- **No warnings when compiled with the following flags:**

  -g -ansi -pedantic -Wall

  50% deduction otherwise.
- **No fatal errors when compiling – otherwise 100% deduction**
- provide your solution as a .zip (**not** .7z, .rar or other formats)
  - name your file Lab4_Firstname_Lastname.zip
    - substitute your names (of course).
- Each file must include the following comment block (with your information) :

```
/*****************************************************************
* Student Name:
* Student Number:
* Course: 23F_CST8234
* Declaration:
* This is my own original work except where sources have been cited.
*****************************************************************/
```

- **Marking Rubric**

  No warnings when compiled with the following flags:

  -g -ansi -pedantic -Wall

  50% deduction otherwise.

- **No fatal errors when compiling – otherwise 100% deduction**

correct, functioning makefile	2 marks
minimum of: .c &.h for each of 3 msg types and main.c	2 marks
Output resembles sample report with aligned headings and columns	2 marks
Input read from stdin, message types separated using scanf() or sscanf() or string.h and stdlib.h functions	2 marks
used arrays of struct, one struct per message type to collect messages	2 marks
Total	**10 marks**