# ECE 275 Assignment 1

**DUE DATE:** Thursday, September 11, 11:59PM

Create a program that will filter a log of temperature data representing internal temperature values recorded within a home. Assuming the temperature data was collected during the summer, the program should determine when the air conditioner was operating by analyzing trends within the collected data.

The assignment name for this assignment is: **tempfilter**

## Command-line Arguments

Your program must be capable of utilizing a command line argument to specify the input log file and output file with the filter data and analysis results.

tempfilter inputFile outputFile

Your program must ensure the user has correctly provided the required command-line arguments and display a *usage* statement if the provided arguments are incorrect.

## Input Temperature Log Format

The input temperature log format will provide a log of recorded times and temperature readings, in which one reading will be specified per line, using the following format:

**HH:MM TT.T**

* **HH:MM** represents the time at which the temperature was recorded and is specific in 24 hour format.
* **TT.T** represents the recorded temperature reported in Fahrenheit as a double precision value (note: all temperatures will be in the range [0.00ºF, 99.9ºF]).
* The time and temperature will be separated by one or more whitespace characters.

The following is a simple example of the input file format:

09:30 71.2

09:31 71.0

09:32 70.8

09:33 70.4

09:34 70.7

09:35 70.6

The temperature log may be missing data and not contain a complete log of temperature for all minutes within a day.

## ACTempData Structure

ACTempData is a data structure for representing AC temperature readings and status. The typedef and struct definitions for the ACTempData data type are:

typedef struct ACTempData\_struct {

double temperature;

bool status;

bool valid;

} ACTempData;

The temperature member stores the temperature reading, the valid member indicates if the reading is valid, and the status member indicates if the AC was On of Off.

For keeping track of the temperature data in your program, your program must use an array containing one day's worth of ACTempData elements, defined within main as:

ACTempData acTemps[MinPerDay];

MinPerDay is a constant defined as:

static const int MinPerDay = 1440;

## Temperature Filtering Part A: Marking Erroneous Data Invalid

Your program should first determine erroneous temperature readings. An erroneous temperature reading is defined as a temperature reading that is either more than 5 degrees above or 5 below the temperature recorded in a previous valid minute. All erroneous temperature readings should be marked as invalid. The first element read in from a file is always considered valid.

## Temperature Filtering Part B: Simple Low-Pass Filter of Consecutive Samples

As temperature sensor readings may be noisy, we would like to apply a simple low-pass filter to filter any sequences of **consecutive, valid** temperature readings. The following defines a simple low-pass filter that can be used for these purposes:

filtered\_output(0) = unfiltered\_input(0)

filtered\_output(n) = 0.9375 \* filtered\_output(n-1) + 0.0625 \* unfiltered\_input(n)

After removing the erroneous data, your program should apply the low-pass filter to sequences of temperature readings for consecutive times. If a non-valid temperature reading is found, the filter should restart at the next valid temperature reading.

***Note:*** A good article on low-pass filter for embedded software can be found in the article: [A simple software lowpass filter suits embedded-system applications](http://www.edn.com/design/systems-design/4320010/A-simple-software-lowpass-filter-suits-embedded-system-applications)

## Temperature Filtering Part C: Trend Extraction

Given the filtered data, your program should analyze the sequence of valid temperatures to determine the state of the AC. If a non-valid temperature reading is found, the trend extraction should restart at the next valid temperature reading. Trend extraction analysis operate as follows:

1. The AC status for the first entry in a sequence of valid reading is always Off.
2. For a valid temperature reading that does not start a sequence:
3. If (1) the previous AC status was Off, (2) the previous temperature is valid, (3) the current temperature is less than the previous temperature, and (4) the next temperature is valid and less than the current temperature, then the AC status for the current temperature will be On. Otherwise, the AC status for the current temperature will be Off.
4. If (1) the previous AC status was On and (2) the current temperature is less than or equal to the previous temperature, then the AC status for the current temperature will be On. Otherwise, the AC status for the current temperature will be Off.

## Filtered Output File Format

The filtered output log should be formatted with one ***valid*** entry per line using the following format:

**HH:MM TT.TTT AC**

* **HH:MM** reports the time at which the temperature was recorded specified in 24 hour format.
* **TT.TTT** represents the filtered temperature value in Fahrenheit as a double precision value with three decimal digits of precision (note: here we assume all temperatures will be less than 100ºF in the house!).
* **AC** reports the determined status of the air conditioner, where a 0 represents the air conditioner was *Off* and a 1 represents the air conditioner was *On*.
* Each entry should be separate by a single tab character (\t).