ENG1008 C Programming
Project 2

This is a group project consisting of **5 students** in each group.

Aims
- To read in sensor data into single and multidimensional arrays;
- To carry out data analysis on it.

Introduction
A small text file (proj2.txt) containing the data (i.e. temperature readings) from the sensor is given. ***To avoid formatting errors, download the proj2.txt file and use it as it is – do not cut and paste into another text file.***

You should use I/O redirection and standard scanf statements to read the readings into the program. I/O redirection forces the program to read the data from the proj2.txt file instead of from the keyboard. The scanf function is being used in the usual manner to get the input, as if the data was read from the keyboard.
To read input from file proj2.txt (instead of from the keyboard) and assuming the compiled program is named a.exe, type:

```
./a < proj2.txt
```

Readings
The data contains sensor readings from 7:00 AM to 8:59 PM at intervals of 15 mins.
There are 30 days of readings per month and three months of data (i.e. April, May and June) are recorded.

Note:
- Global variables are not to be used in the project.
- Only 4 arrays, as described below, should be used in the project.
- You have to use the following lines at the start of the program:

```
#include <stdio.h>
#include <math.h>
```

Your program should do the following:

1. Create FOUR (4) arrays (Single and Multi-dimensional arrays)

1.1. The "readings" array is a 1 dimensional array.
This array is used to store the raw readings read in from the proj2.txt file.

1.2. The second to fourth arrays are 2 dimensional arrays (month arrays) representing the 3 months of April, May and June.

1.3. The first dimension of each array represents the days in each month and the second dimension are calculated readings/values for a particular day.

1.4. The calculated readings/values for each day are:
1.4.1. the hourly average temperature starting from 7:00 AM (i.e. 7:00 AM to 8:59 PM)
1.4.2. this is followed by the mean temperature for that day
1.4.3. the last element is the standard deviation for that day

2. Implement THREE (3) functions

2.1. Function: **`void hourave(parameter-list ...)`**

2.1.1. This function reads the 30 days of data associated with the corresponding month (either April, May or June) from the "readings" array (as specified in 1.1).

2.1.2. It calculates the average temperature in every hour and updates the hourly average temperature to the corresponding day in the month array.

2.1.3. For example, the temperature readings recorded between 7:00 AM to 7:59 AM are used to calculate the average temperature for the first hour (7:00 AM) while the temperature readings recorded between 8:00 PM to 8:59 PM are used to calculate the average temperature for the last hour (8:00 PM) of the day.

2.1.4. This function has the following parameters/arguments:

2.1.4.1. the readings array (as specified in 1.1 above);

2.1.4.2. the month array (either April, May or June array as specified in 1.2);

2.1.4.3. additional parameters may (or may not) be needed to differentiate which month's data is being processed.

2.2. Function: **`void dailymsd(parameter-list ...)`**

2.2.1. This function reads the 30 days of data associated with the corresponding month (either April, May or June) from the "readings" array (as specified in 1.1)

2.2.2. It calculates the mean temperature and standard deviation in a day using all the readings recorded between 7:00 AM and 8:59 PM for that day.

2.2.3. It updates the mean & standard deviation to the corresponding day in the month array.

2.2.4. This function has the following parameters/arguments:

2.2.4.1. the readings array (as specified in 1.1)

2.2.4.2. the month array (either April, May or June array as specified in 1.2)

2.2.4.3. additional parameters may (or may not) be needed to differentiate which month's data is being processed

2.3. Function: **`float monthhr(parameter-list ...)`**

2.3.1. This function will calculate the monthly average temperature for a specified time.

2.3.2. This function has 2 parameters/arguments:

2.3.2.1. the month array (either April, May or June array as specified in 1.2);

2.3.2.2. the hour (0 for the first hour 7:00 AM; 1 for the second hour 8:00 AM, and so on).

2.3.3. This function will return the calculated monthly average temperature for the specified hour.

2.3.4. For example, the hourly average temperature for Day 1 to Day 30 at 7:00 AM (i.e hour specified is 0) will be averaged to give the monthly average temperature at 7:00 AM.

3. Read in the data from the proj2.txt file using I/O redirection and scanf statements.

3.1. For the output,

3.1.1. the program is to print out for each day the hourly average temperatures;

3.1.2. the daily mean, the daily standard deviation and to repeat this for the 30 days in the month;

3.1.3. the program will also print out the monthly average temperature from 7:00 AM to 8:00 PM.

Deliverables

1. Report (PDF format) consisting of Aims, Algorithms and Methods, Testing and Results, Conclusion and Appendix (Program listing).
2. Original source code (*.c format).

Deadline

Submission to LMS Dropbox by end of Week 13.

Assessment Rubrics

	Component	**Weightage**
1	Algorithm This describes the algorithms either in terms of pseudocode or as a flow chart/diagram for the functions **`hourave`**, **`dailymsd`** and **`monthhr`**	20
2	Programming • Style: the program should have the appropriate layout, as shown in the lecture slides. • Generous use of comments to make sure that everything is explained clearly. Comments should be vertically aligned as much as possible. In addition, make sure that the comments do not 'spill over' to the next line – this is very distracting and looks messy.	50
3	Documentation (a) Content: • The format of the report must include the Aims, Algorithms and Methods, Testing and Results, Conclusion and Appendix (Program listing). • Program listing to be included in the Appendix (cut and paste from the source code .c file). • DO NOT provide the source code in the form of screen dumps as this is very difficult to read and impossible to annotate properly. Instead, copy and paste the program from a text editor such as Notepad. (b) Testing & Results: • This would normally be screen dumps of the working program for both valid and invalid inputs.	10 20
	Total	**100**

Feedback procedure

After your assignment has been marked, feedback will be provided.