Assessment Brief

Module title	Procedural Programming
Module code	COMP1711
Assignment title	**Collaborating with HealthMate CIC on a Step Tracker Data Analysis Tool**
Assignment type and description	This is a programming assignment. You will design and create a number of programs to solve a problem from your client.
Rationale	You are doing this assessment to develop your skills in software design and development.
Word limit and guidance	You will create a number of tested and working C programs
Weighting	100%: This is the only assessment for this module
Submission deadline	Part 1: Friday November 3rd 2023 (by 2pm) Part 2: Wednesday 13th December 2023 (by 2pm)
Submission method	Online through Minerva.
Feedback provision	You will get automated feedback through Minerva and Gradescope
Learning outcomes assessed	LO1: select appropriate data types to represent data. LO2: apply problem solving techniques to develop a programming solution to real world problems. LO3: design, implement, debug and test procedural programs.
Module leader	Martin Callaghan
Other Staff contact	Amy Brereton, Max Fasi, Arash Bozorgchenani, Dibyayan Chakraborty

1. Assignment guidance

HealthMate CIC, a Community Interest Company based in the UK, is embarking on an ambitious project to support healthy lifestyles and improve digital literacy among high achieving secondary school students.

They intend to use open-source tools and platforms like the BBC Micro:Bit (as teachers and schools already have extensive experience of the device).

Their aim is to create a **proof of concept** that combines health consciousness with the teaching of digital proficiency.

As intern software developers, you will work on the development of a step tracker data analysis tool, the first step in this large project.

Your mission is to develop a console application in C capable of analysing step data either from a data file, or directly from a Micro:Bit device.

This tool should offer a large range of analyses to give users detailed insights into their daily physical activity, supporting HealthMate CIC's vision of a digitally literate and health-conscious younger generation.

Although you will be expected to submit files to Gradescope for marking, you should continue to use Git and Codespaces to develop your code.

If ever there are doubts whether work is completely your own, we will be able to look at your commit history and this will be very good evidence for you to prove that there is no issue with academic integrity.

We can give you general guidance and remind you where to find relevant supporting material in the notes, but coursework is individual and by submitting it, you are confirming that it is your own individual work.

You **must** attempt the part 1 task. All the part 2 tasks are optional but remember that the passing grade for the module is 40%.

2. Assessment tasks

Part 1: Submission deadline Friday November 3rd 2023, 14:00.

You have unlimited submissions attempts until the deadline.

Total marks available: 15

Task 1: Understanding our data files
Difficulty level: BRONZE

Task 1 marks: 15

You are given a data file called **FitnessData_2023.csv** and a code template file **`StepsTask1.c`**

Each row in the csv file looks like this:

```
2023-09-01,08:15,150
```

- The first column represents the date (in YYYY-MM-DD format).
- The second column is a time (8.15am in our example)
- The third column represents the steps counted in the 15 minutes immediately before the time stamp. Here, it means that the step counter has counted 150 steps between 8.00am and 8.15am.

Your client wants you to write a single C program that will:
- Read in this csv file
- Store it in a suitably sized and structured array and typedef data structure
- Write the number of records in the file to the screen in this exact format (234 is just an example):

  ```
  Number of records in file: 234
  ```

- Write to the screen the first three rows of the file (corresponding to the times 07:30, 07:45, 08:00). This should be in the format:

  ```
  2023-09-01/08:15/150
  ```

  Note that there are **NO SPACES** in the output string.

In both cases, note that there is **NO WHITESPACE** before or after the strings. We will test for this format **EXACTLY**.

The code will be marked automatically, so if your source file does not have the correct name or you get the output wrong it is likely you will get no marks.

We will test it with multiple different files – so don't "hard code" the output.

This will involve you using an appropriate typedef (custom data structure).

```
typedef struct {
    char date[11];
    char time[6];
    int steps;
} FITNESS_DATA;
```

The template program we supply (see Minerva) has this typedef already in it. You can either use this, or you can create your own struct if you prefer.

Also in the template, you will see that we have already written a helper function called **tokeniseRecord** that you can use to split the string representing each line from the file into three individual token strings representing the data from the row.

You do not need to understand how this function works and should not change it in any way. This is the same function that we have used before in our teaching sessions.

When you submit the file it should have the exact filename:

`StepsTask1.c`

How it will be marked:

1. The ability to read in the file we give you without any errors:	2.5 marks
2. The ability to read in another data file that you have not seen:	2.5 marks
3. Printing out the number of records in the file	2.5 marks
4. Printing out the requested rows:	7.5 marks

Part 2: Submission deadline Wednesday December 13th 2023, 14:00.
Total marks available: 85
You have unlimited submissions attempts.

Task 2: Analysing our data
Difficulty level: SILVER

Task 2 marks: 30

Following on from your initial program, you now need to expand your initial program to add some features to analyse the data and help us understand what the data means.

Following good programming practice, you should split your program functionality into two files:

`FitnessDataStruct.h`	– a header file containing the struct typedef and any helper functions you want
`StepCounter.c`	– a program that contains the code and functions to solve the requested tasks

The program should provide a simple menu system and ask the user to select from the following options and an option to quit.

A: Specify the filename to be imported – you need to check that the file opened correctly.
B: Display the total number of records in the file
C: Find the date and time of the timeslot with the fewest steps
D: Find the data and time of the timeslot with the largest number of steps
E: Find the mean step count of all the records in the file
F: Find the longest continuous period where the step count is above 500 steps
Q: Exit

These should be printed out with the format given below:

Option	**Example output**
A	Input filename: **Note – if the file cannot be opened successfully:** Error: could not open file Program **returns 1** & exits
B	Total records: 229
C	Fewest steps: 2023-09-01 14:20
D	Largest steps: 2023-09-01 18:00
E	Mean step count: 427 **Note – this value should be rounded appropriately and printed as an integer.** **For example, 234.2 -> 234** **234.5 -> 235**
F	Longest period start: 2023-09-01 14:20 Longest period end: 2023-09-01 18:00 **Note – the end is the final period where the step count exceeded 500.**
Q	Program **returns 0** & exits

How it will be marked:

1. Providing a menu system providing options A-F and Quit:	5 marks
2. Successfully coping with an incorrect filename:	4 marks
3. Displaying the total number of records:	4 marks
4. Date and time of fewest steps:	4 marks
5. Date and time of largest steps:	4 marks
6. Mean step count:	4 marks
7. Longest continuous period:	5 marks

Ensure that your program has implemented the Q/quit option before testing with the auto-grader as without it, the program will time out and fail.

Task 3: Outputting sorted data
Difficulty level: SILVER

Task 3 marks: 15

This task requires you to write a utility program to output the data file sorted by descending order of step count (so the row with the highest step count should be at the top) in **tab separated values** (tsv) format. A .tsv file differs slightly from a .csv in that the delimiter isn't a comma (,) but a tab character. This has the special code **\t** and creates a tab space between values.

You will be supplied with a template file with the same **tokeniseRecord** helper function you used before.

Your program should:

When you run your program, it should:

1. Ask the user for a filename.
   `Enter filename:`
1. Your program should ensure that the filename provided is a valid file (able to be opened by C), and that all of the data inside the file is in the correct format (for example, 12-15-2023,,200 would be invalid as the time is missing). If the file does not exist or if it contains invalid data, your program should give a suitable error message, **return 1**, and exit:
   `Error: invalid file`
2. If the file exists then your program should read the file into an array, and sort the data in **descending order** of steps (highest number of steps at the top, smallest number at the bottom). If two records have the same step count, these should be consecutive but the order does not matter.
3. Write out the sorted data file with the same filename, but with the file extension .tsv added to it. Once this is complete, return 0 to exit successfully.

If the input file is:
mydata.csv

```
2023-12-30,09:30,598
2023-12-30,09:45,376
2023-12-30,10:00,521
```

The output file will be:
mydata.csv.tsv

```
2023-12-30	09:30	598
2023-12-30	10:00	521
2023-12-30	09:45	376
```

(Note that the spaces are **\t** characters)

How it will be marked:

1. Providing the menu option and coping with an incorrect filename:	5 marks
2. Coping with an incorrectly formatted file (the fields in the wrong order):	5 marks
3. Creating an output file in the correct format:	5 marks

Note that the auto-grader expects the program in exactly this structure- it will enter a filename and expect the program to run without needing any further input.

Task 4: Using the BBC Micro:Bit accelerometer as a step counter
Difficulty level: GOLD

This is an extension "stretch and challenge" task. **We do not expect that all students will be able to complete it.**

Task 4 marks: 25

The BBC Micro:Bit contains a 3-axis accelerometer that you can access using the C driver library we have used in our sessions. Accelerometers are often used in step counters.

Your task is to explore how the accelerometer works and write a C program for the Micro:Bit that will record the steps taken for a single 5 minute interval. Your program should provide some mechanism to allow this value to be accessed or downloaded.

This task will be assessed through a demonstration and short question and answer session with one of the lecturers or teaching assistants. You will need to book a slot to do this face-to-face or alternatively you may record a short (maximum 5 minute) private YouTube video and let us have the link by the final submission date.

How it will be marked:

1. Describing how you have looked at the data from the 3 axes and decided what might represent a step count (your experimental approach): 10 marks
2. Writing and testing a C program that records the steps for 5 minutes: 10 marks
3. Providing some mechanism to download or access the number of steps: 5 marks

Task 5: Understanding code quality
Difficulty level: BRONZE

Task 3 marks: 15

For this task you will need to complete the multiple-choice quiz in Minerva. You will have **one** submission attempt at this, and you will have **60 minutes** to complete and submit your answers.
You will do this in your own time, and we suggest that you should use a computer with a stable internet connection and find a quiet place where you can complete this alone without distractions.

This is an assessment, and therefore must be completed independently. You can use any notes or online resources you would like, but you should not be working with other students.

It has 15 questions related to good programming practice.

How it will be marked:

Each of the 15 questions is worth 1 mark.

3. General guidance and study support

Please refer to the module teaching materials in Minerva for any background support.

4. Assessment criteria and marking process

This assignment tests the Learning Objectives for this module:

Marks and feedback will be returned to you approximately three working weeks after the final submission date.

The passing mark for the assessment is 40% and this can be gained from any task or combination of tasks.

5. Presentation and referencing

You should produce working and tested C programs, uploaded to Gradescope with plenty of comments to explain your work. We also expect you to use meaningful variable names and for code to be indented to make it readable.

If you use an idea in your solution that you have seen on the Web somewhere then add a comment to the code like this:

```
// I based this on an idea I found here:
// https://stackoverflow.com/questions/12901021/error-in-c-file-handling
```

If you use a book or a printed resource then give the author and title:

```
// I based this on an idea I got from here:
// An Introduction to C and GUI Programming by Simon Long, pages 56-57
```

If you used ChatGPT, the tell us the prompt you used:

```
// I based this on a discussion with ChatGPT
// Prompt: Explain how to import a CSV file in C
```

6. Submission requirements

All code should be uploaded to the submission points in Gradescope (we will explain fully how to do this in our sessions).

7. Academic misconduct and plagiarism

Leeds students are part of an academic community that shares ideas and develops new ones.

You need to learn how to work with others, how to interpret and present other people's ideas, and how to produce your own independent academic work. It is essential that you can distinguish between other people's work and your own, and correctly acknowledge other people's work.

All students new to the University are expected to complete an online Academic Integrity tutorial and test, and all Leeds students should ensure that they are aware of the principles of Academic integrity.

When you submit work for assessment it is expected that it will meet the University’s academic integrity standards.

If you do not understand what these standards are, or how they apply to your work, then please ask the module teaching staff for further guidance.

By submitting this assignment, you are confirming that the work is a true expression of your own work and ideas and that you have given credit to others where their work has contributed to yours.

8. Assessment/ marking criteria grid

Task	Description	Marks available
Part 1- task 1	Importing a file into an array of structs and printing first two rows	15
Part 2- task 2	Analysing our data	30
Part 2- task 3	Outputting sorted data	15
Part 2- task 4	Using the BBC Micro:Bit accelerometer as a step counter	25
Part 2- task 5	Multiple choice quiz	15