**Module:** C++

**CA:** CA1 - Assignment

**Value:** 10% of Module

**Due Date:** See Moodle

**Objectives:**

* To practice the design and implementation of a C++ based application
* To gain experience with the use of C++ data structures (structs)
* To gain experience in using C++ containers

**Core Requirements**

**Stage 1 (Data-Generation)**

Create a CSV file containing at least 100 rows of data on any topic you wish. The topic you choose should have a least 5 fields - including a combination of strings, integer and floating point numbers.   
Initially, read in the lines from the file and display them on screen using neat formatting.  
***You may use online data generation tools to generate your data e.g. mockaroo.***

**Stage 2 (Data-Loading)**

1. Create a **struct** to represent a row of data from your CSV file. **E.g. movie**   
   Appropriate data types should be used for each field in the struct definition.
2. Read in the data from your CSV file into a *vector of structs.***E.g. vector<movie>**This should be implemented in a function that accepts a reference to a defined vector, and uses the reference to populate the vector.

**Stage 3 (Analysis)**

1. Write a function that will accept the vector of structs, iterate through the elements, and **display the elements** in a neatly formatted fashion.
2. Write a function that will accept two parameters: the vector of structs and a value for a particular field (of your choosing). The function **will search through the vector elements for the value** (in a particular field) and will **return the index (int)** of the located element,   
   or -1 if no match is found. Use the returned index to access and display the contents of the struct, otherwise, display a not found message.
3. Write a function that will accept the vector of structs, iterate through the data loaded in stage 2 and accumulate a **count of the number of rows that are a match for each of the unique values** in a chosen column. For example, if you chose *movie* data then this function could return the number of movies of each genre (comedy, thriller, drama, etc.). This function should **return a map** containing all information.
4. Write a function that displays a **subset** of the data rows **based on a user input**. For example, this function could show all of the movies of a particular genre entered by the user and passed into the function as an argument.
5. Write a function that will find the **highest, lowest and average of a discrete numerical field (Integer).** The function should **return the average value** (as an int) **and provide (“return”) a copy of the structs with the highest and lowest values.** For example**,** if you chose movie data then this function could provide the highest rated and lowest rated movie, and also return the average rating of all movies.
6. Write a function that will search through the data and **return a *list*** of all items that match **or partially match** a given text input.For example, all movies that contain the text “The” in the title. **Iterators must be used for this function.**
7. Write a function that will display all of the data in **descending order** of a selected **floating point** field.For example, using the **gross takings** of a movie in Millions (Floating point number), you could display them in descending order of gross takings.

**Stage 4 (User-Interface)**

Create a **console-based menu system** which exposes the functions developed in stage 3 to the user.

* All results must be clearly displayed.
* Tables should be correctly aligned.
* Floating point numbers should be displayed to two places of decimal.

**Repository & Design Requirements**

You are required to create a GitHub student repository for your project. This project must have the following properties:

* Name format (i.e. “CPP\_2025\_CA1\_*student-initials*”)
* The project repository must be shared (read access) with your lecturer(s) on the module.

**If your project does not show clear evidence of a substantial, consistent and evenly spread Git/GitHub commit history then your project will receive a mark of zero percent.**

Where a project has an acceptable Git commit history, it will awarded a maximum of   
**5 marks** based on the appropriatness of the commits and the quality of comments (clarity, accuracy, descriptiveness, use of illustrative emojis).

### **Deadlines**

Late assignments are subject to the DkIT Continuous Assessment policy, as outlined here

<https://www.dkit.ie/about-dkit/policies-and-guidelines/academic-policies/student-centred-learning-teaching-and-assessment/continuous-assessment-procedures.pdf>

Extensions will only be considered where there are **legitimate verifiable reasons** which are communicated to the lecturer prior to the submission deadline with the following form.

<https://www.dkit.ie/about-dkit/academic-schools/school-of-engineering/documents/school-documents/engineering-continuous-assessment-absence-form.pdf>

The following will apply in the event of late submissions without the above completed form

1. Material submitted four weeks after the due submission date or after the start of the examination period at the end of the relevant semester will not be marked.
2. The marks awarded to the assessment element will be reduced by 20% for material submitted up to one week following the submission date. (e.g. a essay submitted four days late which had a face value of 60% would be given 48%, a piece of work of value 45% would be allocated 36%)
3. If an assignment is overdue by more than one week but is submitted within four weeks of the due date it should be marked as (b) above, but with a maximum mark attainable of 40%

**Submission Requirements:**

1. **All Source code must be submitted via Moodle in a zip file. This must include a README containing a single link to your GitHub repo.**
2. The assignment **must** be entirely the work of each student. Students are **not** permitted to share any pseudocode or source code from their solution with any other individual or group in the class. Students may **not** distribute the source code of their solution to any student in **any** format (i.e. electronic, verbal, or hardcopy transmission).
3. Plagiarised assignments **will receive a mark of zero**. This also applies to the individual allowing their work to be plagiarised.
4. **The use of generative AI tools is strictly forbidden.**
5. Any plagiarism **will** be reported to the Head of Department and a report will be added to your permanent academic record.
6. Each student **must** complete and sign a single assignment cover sheet. Please submit this with your Moodle submission.
7. An **interview** will normally take place to assess the student’s work on this project. **Failure** to attend this interview will result in a grade of **0%** in this component. Your final grade is **directly related** to your performance in this interview. Each student will normally be asked several **questions** related to the **concepts** covered in, and their **contribution** to, the project. Failure to adequately **answer** a question will result in a **penalty** applied to the final mark for that student.