CST2550 Coursework 2

Dr Barry D. Nichols

December 20, 2022

1 Brief Task Description

A new audio streaming service requires a system to maintain the library of tracks available. There will be billions of tracks in the system and they will be constantly searched by artist/band name. New tracks will also be added regularly by loading a list of new tracks from a file.

You should select, with justification, and implement an appropriate data structure to store song details to be searched. You must complete time complexity analysis of the key operations using this data structure. A sample file of song details will be made available for use in designing, implementing and testing the system [note: different files of the same format will be used for marking].

The focus of this task is the library and search functionality, you should not implement any audio playing functionality.

You should not use any third party libraries or code as a part of your solution and all code should be written by you, i.e. not automatically generated. You also shouldn't use non-standard or operating system dependant libraries in your program or operating system calls (the code should all be standard C++).

As the focus of this task is the data structure, you must design and implement the data structures yourself. Marks will not be allocated for the use of standard template library (STL) data structures.

Apply the knowledge you have gained on this module to select and design the data structures and algorithms which are most appropriate and to analysis the time-complexity for your report.

2 Submission

You must submit a single **zip** file of all required source code; a single **PDF** report and an **MP4** video demonstration of the software by **5pm on Friday, 21st April 2023**.

The source code zip file should include:

- C++ source code files of your program
- catch2 test code
- a Makefile to compile your program

The report must be no longer than 7 pages (including references), with a font size of 11 or 12 point. If your report exceeds 7 pages only the first 7 pages will be marked. This is a generous limit and you should not need to write an 7 page report.

The video demonstration should be short and is **not a presentation**, it should clearly demonstrate your knowledge of the implementation. There is a strict 5 minute time limit and only the first 5 minutes of any longer videos will be used in marking your work. This is a generous limit and there is no requirement to use all 5 minutes.

Note:

- **To allow anonymous marking your report and zip file must be named using your student ID, i.e. M00123456.zip and M00123456.pdf**
- **Only source code (header and cpp files) and a Makefile should be included in the zip file, no other files**
- **If code does not compile and run it will severely limit your marks for the code**
- **As anonymous marking will be applied, you should *not* include your name in your source code or report**
- **No marks will be awarded for code without a demonstration video.**
- **Marks for code will be severely limited if you do not demonstrate understanding in the demonstration video**

3 Scenario

A new audio streaming service will be introduced. At this initial phase of development a program with efficient track search functionality is required. **The client program with audio play functionality is NOT required at this stage**.

Details of tracks will be loaded from files. If a file is loaded it should add all tracks to the library. If a track is already in the library it should not be added again.

There should be functionality to:

- add all tracks from a file
- save tracks in the library to a file
- search by artist/band name
- remove specific track(s)

Your program must use an appropriate data structure to store the details of tracks in the library.

You will be provided with a small sample data file in the correct format to use when writing and testing your program. However, the program should work with any file of this format (and another file will be used when marking your work); therefore, your program should allow the user to specify the input file name (**do NOT hardcode the file name**).

4 Detailed Description

It is recommended that you complete the tasks in the following order as the later sub-tasks will require the earlier ones.

4.1 Set up Project

Create a Git repository and makefile for the project, remember to update the makefile and commit new files to the git repository as you implement/update them.

4.2 Plan Software

Design and implement a class to store the necessary details of an audio track.

4.3 Design Data Structure/Algorithm

Design the data structure(s) which will store the **Track** objects and associated algorithm(s) to add and remove them. Analyse the time-complexity of the algorithms (this will be needed in your report). You must also justify your selection of data structure, based on the requirements of the program.

You must use **pseudo code** in your design, marks will not be awarded for C++ code in the design section.

4.4 Implement Data Structure/Algorithm

Implement the data structure(s) and algorithm(s) you designed and analysed for the storing and searching of audio tracks in the music library.

4.5 Test Data Structure

Apply software testing (using catch2) to ensure your *Track* class, data structures and algorithms are all working correctly.

4.6 Read Sample Data File

Implement the program to read the sample data from the given file and process it using the data structure(s).

4.7 Implement Menu

Implement an intuitive (command line) user interface to allow users to achieve the required functionality as listed above.

4.8 Write Report

Finish writing the report, including the time-complexity analysis and test cases you prepared while designing the software.

5 Report Contents

The report should have the following sections:

- Introduction
  - brief description of the project (not just repeating the coursework task)
  - a paragraph describing the report layout
- Design, including:
  - justification of selected data structure(s) and algorithms
  - analysis of the algorithms which provide the key functionality (using **pseudo code**, **not C++ source code**)
- Testing, including:
  - statement of testing approach used (not test code)
  - table of test cases (not test code)
- Conclusion, including:

    - summary of work done
    - limitations and critical reflection (including what caused the limitations)
    - how would change approach on similar task in future (should avoid repeating mistakes)

- References
    - in Harvard format
    - must have matching in-text references

6 Video demonstration

The video should be a quick demonstration of the software demonstrating your **understanding** of the implementation (**just showing you know how to use the software will not demonstrate any understanding**).

Run the program and explain how the functionality was achieved. You may show both the program running and the source code, but don't just step through the source code line-by-line reading it out (**this also doesn't demonstrate any understanding**). Just explain key parts of the code which achieve the functionality you are showing.

Marks for code will be limited if your demonstrated understanding of the code is limited.

7 Formative Feedback

You will have the opportunity to submit a draft version of your report by the end of week 21. If you submit a draft in week 21, some written feedback will be available by the end of week 22 stating what you should improve before the final submission. There will be no mark for the formative submission, but the feedback given will help you to improve your work (and your final mark).

8 Academic Misconduct

This is individual work and you should complete it yourself. You should not work as a group and each submit the same work (even with minor changes) as your own. Any material or ideas found online, in textbooks, etc should be properly referenced.

You should familiarise yourself with the university's academic integrity and misconduct policy: https://www.mdx.ac.uk/about-us/policies/university-regulations

9 Extenuating Circumstances

There may be difficult circumstances in your life that affect your ability to meet an assessment deadline or affect your performance in an assessment. These are known as extenuating circumstances or 'ECs'. Extenuating circumstances are exceptional, seriously adverse and outside of your control. Please see link for further information and guidelines:

https://unihub.mdx.ac.uk/your-study/assessment-and-regulations/extenuating-circumstances

10 Marking

The report and included code will be marked according the to attached marking scheme. Marking will be **anonymous**, i.e. the marker will not see the name of the student while marking.

11 Feedback

Provisional marks and written feedback will be available on Moodle within 15 working days of your submission. If you would like clarification or more detailed feedback on your coursework contact your module tutor.

12 Marking Scheme

12.1 Report

Item	**Marks**
Introduction (description of the project, not the coursework task)	5
Appropriate selection of data structure	10
Analysis of data structure and algorithms	20
Conclusion, including: - summary of work done - limitations and critical reflection - how would change approach on similar task in future	5
References (Harvard style)	5
Layout and clarity of report	5

12.2 Code

Item	**Marks**
Code quality (follows code guidelines and user input validation)	10
Makefile	5
Implementation matches design	15
Implementation meets requirements	15
Catch2 test code (quality and matches testing approach in report)	5