CSCE 2211: Applied Data Structures
Term Project Guidelines

The term project is **NOT OPTIONAL**. The project grade represents **20%** of the total course grade.

Guidelines

- The project is intended to be a teamwork
- A group of **3–5** members is to complete the project by the deadline and report the work in a project report (format attached).
- Group members can be from different sections.
- The project group can be the same as that presenting the term paper.
- Names of group members should be registered in the Google sheet on Canvas by **Thursday the 2nd of October, 2025**. The groups cannot be changed past that date.
- Names of group members must be included in the final report with no allowance to change the names from those sent before.
- The final project report should be submitted on Canvas and the **GitHub repo** (please accept the assignment on GitHub through the link on Canvas).
- Final date of submission of the complete project report is **Thursday the 4th of December, 2025**. There will be **NO late policy** for the submission of the project. No submissions will be accepted after the due date.

Academic Integrity

Students are expected to commit to the principles of academic integrity. Any plagiarism detected will result in zero grade for the project. Please also check the course' policy on the use of AI. The code for the project should be written by you, not generated by AI.

Check AUC Academic Integrity

Project Milestones

- Milestone 1 Data Structures (**Monday 27th of October**): The first phase of your project is to build the code for the data structures that you require for the project. You **cannot** use the STL library or any external library as the basis for your project, but must implement your data structures. The implementation should also have proper documentation for the usage of the data structures.

- Milestone 2 Application leveraging the data structures and design documentation (**Monday 17th of November**): The second phase of your project is building an application that leverages your designed data structures for a useful application. The specifications for the deliverables can be found in the grading rubric.
- Final Submission and marketing presentation (**Thursday 4th of December**): This includes the final submission of the code on GitHub, the final report, and a presentation with a marketing perspective for your project highlighting why it is useful and the advantages of your implementation for it.

Grading Rubric

Everything will be individually graded, and all milestones will contribute to your grade.

- Data structures (25%): The data structures used in the project. The project should use non-linear data structures. You can use linear data structures, but the project cannot be made up of only linear data structures. You must implement your own data structures.
- Application (25%): The code for the application leveraging the data structures. Your application should be well designed. The application must include a user interface (i.e., the user must be able to interact with the program even if just through the command line interface).
- Presentation (15%): Final presentation of the project. This includes a working demo of the project and a presentation that covers the choice of the data structures, application, implementation challenges, experimental results, and marketing of your project (marketing your project does not mean making false and exaggerated claims about it).
- Report (10%): Report that covers the aspects in the suggested format, submitted before the deadline. The report should be well written, with appropriate in-text citations and references (Please use a well-known citation style and check automatically generated references for correctness).
- Code and documentation (15%): Your project development should use GitHub from the start of the project. Your code should be functional, well structured, and well documented.
- Individual evaluation (10%): based on the interviews for the three milestones. Every member of the group should be able to explain the entire project, the decisions made throughout the project, and the way the code works.

Suggested Project Ideas

The team may choose to do one of the suggested projects. The team may also come up with a project different from those suggested. In this case, the project should exhibit the use of Data Structures and Algorithms relevant to the course and should serve as a useful application.

1. **Time-range leaderboard query system:** To provide a high-performance, scalable solution for real-time leaderboard analytics, enabling quick retrieval of top scores over user-defined time ranges without scanning the entire dataset. The system should use a data structure like segment trees to enable insertions, updates, and fast retrieval of scores.
2. **Map navigator:** Use a graph data structure to find the shortest path between two cities. This could involve implementing algorithms such as Dijkstra's or A* algorithm. You can use a dataset similar to the one found at this link.
3. **Interval conflict detector:** real-time system designed to identify and flag overlapping time intervals in scheduling and resource allocation applications. By leveraging an Interval Tree data structure, the system efficiently detects conflicts as new intervals are added or modified, ensuring that scheduling remains consistent and error-free. It should handle conflicts across multiple rooms, devices, or personnel schedules, and give an alert when an overlap is detected.
4. **File zipper:** Compression of **text** files using Huffman trees. This should work for compressing and decompressing text files. The program needs to work without saving the Huffman tree in an external file.
5. **Text concordance:** Finding the frequencies of characters or words in any text document using self-balancing trees.
6. **High-performance domain name lookup using reversed trie:** This project implements a reversed trie (prefix tree) to store and efficiently query domain names for longest-suffix matches, a crucial operation in DNS routing, content filtering, and policy enforcement. By reversing domain labels before insertion, the data structure transforms suffix matching into a prefix traversal problem, enabling O(L) query time, where L is the number of labels in a domain. Domains are stored in reverse order of labels (e.g., www.mail.example.com → com → example → mail → www) for direct suffix matching. The program returns the most specific rule/policy applicable to a given domain.
7. **Snakes and Ladders game:** Using the graph algorithm of Breadth-First Search (BFS). You can have a minimal representation of the game in the CLI. You should show the progress of the game following the solution found.
8. **Sudoku solver:** Generating a Sudoku game and its solution by Backtracking. You can have a minimal representation of the game in the CLI that the user should be able to play. The generated sudoku must have a unique solution.
9. **Detection of repeated and unique substrings:** Using suffix trees to detect repeated substrings and those that are unique.
10. **Converter of expression notations:** Create a program that can convert from any input notation to the others, so for example, given an infix expression, it should be able to produce the corresponding postfix and prefix expressions. (Do not use source code from assignments and exercises).

CSCE 2211 Term Project Spring 2025

Suggested Report Organization

Abstract:

This is just a suggested format. You may change or modify it in the way that suits the project work

Keywords:

Some keywords...

1.1 Introduction

(Introduce the topic of the project here)

1.2 Problem Definition

(Present here a more detailed definition of the topic or problem)

1.3 Methodology

(Outline the methodology chosen to solve the problem)

1.4 Specification of Algorithms to be Used

(Describe the algorithms you will use, you can add your pseudocode...)

1.5 Data Specifications

(Specify the input data to be used in your work)

1.6 Experimental Results

(Give your results of processing the input data in text, tabular or graphical forms)

1.7 Analysis and Critique

(Your own analysis and critique of the output results and the methodology/algorithms used)

1.8 Conclusions

Acknowledgements

(Acknowledge other people who helped you in producing the project)

References

(List here the references you used to produce the project)

Appendix: Listing of all Implementation Codes
