CC215 Data Structures Project Ideas:

Responsible use of online resources:
Please cite the web sites that you download examples and resources from. Copying partial contents from the Internet is alright to learn from. If you submit fully copied projects, you will receive a zero mark. Please add more features, or change the behaviour of the function, or the interface at least. Please remember that you will display a demo of your project running and explain how you wrote the source code. If the project doesn't run, you will loose a lot of marks. Similarly, if you fail to explain how the source code was written, you will loose marks.

General Rules:

- A group can be formed from 3 or 4 students per project.
- Groups are not allowed to repeat ideas, so first come first serve bases, once a group comment on the project's thread with a choice, the idea can not be chosen by another group.

Grading Criteria:
The project is worth 10% of your final mark. These marks are graded individually by asking each student in the presentation for their contribution and testing their understanding. These are broken into:

- 5 marks for correctness (no compilation or run-time errors).
- 4 marks for the application of course concepts, where feasible, (such as: lists, stacks, queues, trees, traversals, and graphs)
- 1 marks for the interface, presentation, documentation, and teamwork.
- 5 bonus marks for all other concepts you self-study outside the learning objectives of the course.

Submission Details:

All project files are zipped and submitted named as "proj_LeaderStudentID.zip", where "LeaderStudentID" is replaced by a leader team member chosen by all team members. The zip file should contain:

- All source code used to develop the project, either as a jar file, or a folder of the project packages and files.
- A written report (soft-copy and hardcopy) according to the instructions written in the guide and example shown on moodle.

Note:
- Your project will be rejected if it does not comply with the above submission requirements.
- When you submit your project you get 5 marks until you prove that it is your own work, during a demo in Week 15.

1. **Project code: CC215-01: Mini Search Engine**

The search engine should be able to "quickly" find out all occurrences of a phrase in a set of 50 given text files written in English. Use any data structure and algorithm of your own choice.

2. **Project code: CC215-02: Event Scheduling**

Given a set of N events, e.g., events in college festival, each student is asked to choose M events of his/her choice, in order of decreasing preference. If there are K slots available to schedule the events out of which at most P slots can be held in parallel, output a schedule that maximizes the happiness of most students. You are free to choose the definition of "happiness" as long as the definition is reasonable. The algorithm should work for any given N, M, K, P and the preference list of any number of students.

3. **Project code: CC215-03: Simple Computer Game**

Design a human vs. computer game of your own choice. The game should be nontrivial enough so that the computer is at least able to beat a few intelligent first-time players

4. **Project code: CC215-04: Text File Compression**

The compressor should be able to compress text files written in English, and the compression ratio (ratio of the original file size to the compressed file size) should be comparable to the freely available file compression software. Use any algorithm and data structure of your choice.

5. **Project code: CC215-05: Spell Checker**

The spell checker should be able to "quickly" find out all spelling mistakes in a given text file written in English. Use any data structure and algorithm of your own choice.

6. **Project code: CC215-06: Origami or Paper Folding**

Simulate any interesting 3D shape that is possible to be created from a piece of paper by following the steps used in your algorithm. The creation should be visually or intellectually appreciable.

7. **Project code: CC215-07:** File-System Search

The system should be capable of searching for files in a large file- system. Some key tasks specific to this project are:

- Specifying application requirements in detail, with attention to details such as the data that is to be searchable, the query language used for searching, the user interface, and the performance constraints (e.g., response time for queries, running time and load for background indexing, and space used by indexes).
- Identifying existing applications that are similar to the proposed one, with particular attention to identifying similarities and differences.
- Identifying standard data structures used for indexing various data types required by the application and evaluating their effectiveness for this application, using a combination of methods (prior results, experiments, etc.).
- Implementing the complete application using a combination of data structures and related methods.
- Experimentally evaluating the performance of the application on real (or realistic) inputs.