CS201 Spring 2019 Portfolio Project

You will create an application that uses algorithms and data structures to sort and search a large dataset or a game that requires a graph-based search. Projects will have minimum requirements that must be met to be graded.

Option 1: Million, billion, trillion…: You will load and provide CRUD access to a large dataset

Data set options:

- Nutrition Tracker-create and manage a diet.log using the USDA branded food item database (https://ndb.nal.usda.gov/ndb/)
- Movie catalog-create and manage a move catalog using the IMDB database (https://www.imdb.com/interfaces/)
- Custom dataset must be submitted by March 4th and approved by March 8th
    - Must be over 100MB and over 10,000,000 records, politically agnostic
    - Proposal includes source/content/size of dataset and what application you are proposing

Each data project must:

- Load the lookup dataset into memory or create an in-memory index to the (unsorted) files. The lookup dataset cannot be advantageously sorted.
- There should be a user diary (nutrition) or catalog (movies) that a user can create, retrieve, update, and delete (CRUD) records. Multiple users are denoted by multiple datafiles (i.e. monica.log would be the log for user monica). Users should be allowed to select the appropriate data file to continue work at a later time.
- The user interface can be ascii based. Make sure that you plan for input errors and provide command help. Curses can also be used. Any other interface approaches need to be submitted by March 4th and approved by March 8th.

Option 2: Playing Games: You will create a graph-based game

- Connect-Four with arbitrary board size-one player against computer and player against player-uses a function to determine wining state and must use a graph to determine the "next best" move
- Boggle with arbitrary board size-one player game with computer calculating all possible word combinations-(uses dictionary file to check words or select words)
- Scrabble with arbitrary board size-one player against computer and player against player (uses dictionary file to check words or select words)
- Custom game - Must use BFS or DFS on a large graph and must be submitted by March 4th and approved by March 8th

Each game project must:

- Consist of ascii or curses-based interfaces
- Players should be able to choose game mode (single, double or playing against computer).
- A series of matches can be played between the same opponents. The program should keep score.

General Requirements

- If you want feedback on your project, you must submit a project outline by March 4th. Outline can include user interface designs, sequence diagrams, and/or record structures. Code will not be evaluated. No feedback will be given after March 8th.
- The user interfaces will be complex with multiple modes and operations. Making the state clear will be important and will take some thought. This is an opportunity to be creative and not be constrained by low expectations. Each project will have different challenges.
- All projects will be developed and hosted in GitHub (think history) and the downloaded zip file will be the submission. The repository will be made public at the time it is due (March 29th)
- All projects will run on Ubuntu, written in ANSI C11 and will use only standard libraries (obviously no STL). Any example code that is used must be documented/cited in comments in the source code. Less than 30% of your code can be "borrowed". This means your code structure should not score high on the software similarity scale when compared to another student or any code available online (see https://theory.stanford.edu/~aiken/moss/)
- All projects must include source code, data download links (too big to host in github), instructions (Readme.md) and a video demonstration. Projects must function per the specifications and the project rules. Applications that throw errors or provide incorrect output will not be graded.
- Projects will be graded for:
  - UI ease of use, algorithm design and efficiency, code readability and modularity, creativity, simplicity, robustness, documentation, etc
- The top 10% in each project category will receive 100%, above average submissions will receive 85% and projects that meet minimum requirements will receive a 70%. All other projects will not be graded and will receive a 0. **NO LATE PROJECTS WILL BE ACCEPTED.**
- Be mindful that other students will be assigned to critique your code. You will need to be open to non-personal feedback. You will critique the code of others. Your professional, constructive, timely feedback will be graded and can earn you up to 10% additional points on the project.

*** Additional direction on projects will be given at the beginning of class. Students will have the opportunity to ask questions at that time. Further clarifications will be made via blackboard. Students will be responsible for updates given either in class or via blackboard.