Lab 1: Why Databases?

Initial decisions

- Initially thinking through this problem, I am given a couple different choices
    - Programming language
        - What language makes parsing text files the easiest?
        - What language makes command line prompts easy?
    - Implementation
        - How do we store and parse data?
    - Project structure
        - What is the structure our project will operate on
- Decisions
    - I will be using python for its easy text parsing abilities
    - Project structure will flow through a single function, main, which handles the instructions the user provides and passes them off to a subfunction to handle specific searches.
        - Subfunctions:
            - search(entryPosition, entry): a search function that returns any lines the entry at entry position is found (in an array)
            - return_line(line): returns the data on some line in the form of an array
            - Also, there will be specific functions for each query option relying on the previous two functions.
        - The reason why I chose this architecture is that it separates each task into a function, making testing easier.
    - Dev environment: using VSCode and testing on Calpoly's unix servers.

Task Log

- search(entryPosition, entry)
    - Approximate time requirements: took around 10 minutes to set up this first function and the basic file structure.
    - Implementation: Processes text file line by line, searching at entryPosition. If entry is found, the line number currently being processed is added to a list and returned. If no data is found, [] is returned
- return_line(line)
    - Approximate time requirements: took around 5 minutes to set up this function.
    - Implementation: simply returns the line at line number line
- return_entry(line, entryPosition)
    - Approximate time requirements: took around 3 minutes to set up this function.
    - Implementation: Uses return_line and returns just the data at entryPosition.
- main()

    - Approximate time requirements: took around 5 minutes to set up this function.
    - Implementation: The main function orchestrates a loop that continuously asks the user for instructions until the quit command is received, for which it ends the loop and the program.
- Basic query functions
    - Approximate time requirements (per): took around 5 minutes each for these functions.
    - Approximate time requirements (total): took around 30 minutes for all of these functions.
    - Implementation: the implementation of each of these functions is obviously different, but the pattern is the same; use the search function to look up entries by some parameter, and then format and print its response.
- Average, highest, and lowest gpa functions (+info)
    - Approximate time requirements (per): took around 10 minutes each for these functions.
    - Approximate time requirements (total): took around 40 minutes for all of these functions.
    - Implementation: the implementation of each of these functions is obviously different, but the pattern is the same; use the search function to look up entries by some parameter, calculate some value using the returned data, and then format and print its response.

Testing

- Bugs
    - Search not returning entries when the user doesn't add capitalization to names
        - Lowercase both the user's string and the current entry's string
        - Time requirements: 2 minutes
    - Query functions printing names with the user's capitalization
        - Use the string from the return_line function instead
        - Time requirements: 2 minutes
    - First names being printed before last names
        - Reverse in code
        - Time requirements: 2 minutes

Final notes

- Overall the project took around 2-3 hours in total
- It was a pretty good review on python

Useful links

- https://github.com/TylerHBE/CSC365-Lab1