CSC 365: Lab 1 writeup

Team members: Hannah Moshtaghi (hmoshtag), Kevin Woodman (kwoodman)
Fall 2024

Initial Decisions:

We initially chose Python as our language of choice because it was familiar to us and we figured we could make use of some of its built in libraries to make our task a little bit more straightforward. We both used VS code for our IDE, and conducted our collaboration on this lab with GitHub.

Chosen Internal Architecture:

To tackle this lab, we utilized several data structures to store student data. For instance:

- **Arrays**: The primary data structure (studentArray) is a 2D list derived from student.txt where each sublist represents a single student. Each of these sublists contains multiple attributes such as the student's first/last name, grade, bus, etc. All of these attributes are represented as strings.
- **Dictionary**: grade_count within the displayGradeCount function is a dictionary/hashmap that is used to store the count of students in each grade. The grade is the key and the value is the number of students in that grade. A dictionary was useful for this function because it prevents us from looping over the list multiple times when keeping track of numbers for various grades.

On another note, we made use of integer constants to represent specific fields in the student sublists. For example, LASTNAME = 0, FIRSTNAME = 1, etc. These are used as index pointers to make it easier to reference specific fields in the student sublists.

Task Log:

Task	**Person in Charge**	**Time taken**
Start/Quit Program (**R1 & R12**)	Kevin	20 min
S[tudent]: <lastname> (**R4**)	Kevin	10 min
S[tudent]: <lastname> B[us] (**R5**)	Hannah	15 min
T[eacher]: <lastname> (**R6**)	Kevin	10 min
G[rade]: <Number> (**R7**)	Hannah	10 min

B[us]: <Number> (**R8**)	Kevin	5 min
G[rade]: <Number> H[igh] or G[rade]: <Number> L[ow] (**R9**)	Hannah	30 min
A[verage]: <Number> (**R10**)	Kevin	15 min
I[nfo] (**R11**)	Hannah	10 min
Write up	Hannah	30 min
Testing	We each tested our own functions	Kevin: 20 min Hannah: 1 hr

Notes on testing:

When we finished implementing a given function, we implemented around 3 test cases to cover its basic functionality, testing the short/long form commands and covering edge cases like when an empty line is returned. Time estimates are shown in the table above. Since we tested as we went, it was easier to catch bugs early.

Most of the bugs we encountered involved the typical typos, however, there were a couple that were more long-winded. For example, when first writing the searchByGrade function, we realized that we needed to have a check for if grade_students (the array that holds all the students for a given grade) was empty to prevent the program from crashing given the edge case where you enter a grade with no students. Bugs took a few minutes to ten minutes to figure out.

Final notes:

Overall, this lab helped us gain an appreciation for the databases and query languages that help streamline this search process. We were dealing with a relatively small dataset, which made it easier for us to check our work, however, this task would have been a lot more unruly had we been given a much larger dataset, and may have had to further optimize our code for time complexity.