Lab 01 Part 01 Writeup

Team Members

Kellie Banzon, Cole Bemis, Tanner Larson

Initial Decisions *(programming language, environment)*

We elected to use Python for this lab since we all have prior experience with the language and it supports simple Student objects and lists of objects.

We each selected our own coding environments. Cole prefers to use Vim, while Kellie and Tanner favor the PyCharm IDE.

Internal Architecture *(what data structures you used for what purposes)*

We chose to organize the list of students into several dictionaries that group students by various keys. For example, we have a dictionary called “grouped_by_grade” that maps a grade number to a list of students in that grade. While these structures add initial time complexity and require more storage space, they allow for O(1) lookups when the user enters a query. Organizing the data in this way makes queries like “Given a bus route, find all students who take it” and “Find all students at a specified grade level” trivial to answer. Although some of our dictionaries hold redundant information, we reckoned that the “size” of this database renders these duplicates largely inconsequential. We chose to sacrifice storage efficiency to achieve fast responses to user input.

Task Log

Task	Total Hours	Subtask / Description	Times	Assignee
Internal Architecture	1.00	Created Student class, parsed file into groups of student objects	2019/04/03 10:00-11:00	Cole Bemis
Parsing	1.50	Handle student, teacher, and info queries	2019/04/03 17:30-18:00	Kellie Banzon
		Handle grade, average, and bus queries	2019/04/05 10:10-11:00	Kellie Banzon
		Error handling	2019/04/07 23:20-23:30	Kellie Banzon
Queries	.83	R7	2019/04/03 17:40-17:50	Tanner Larson
		R4, R5, R6, R8	2019/04/03 17:50-18:00	Cole Bemis
		R9, R10, R11	2019/04/08 13:20-14:10	Tanner Larson
Testing	1.5	Writing test suite	2019/04/08 10:40-11:00	Kellie Banzon, Cole Bemis
		Writing R7 test case	2019/04/08 14:25-14:35	Tanner Larson
		Writing R9, R10, R11 test cases	2019/04/08 14:55-15:20	Tanner Larson
		Writing R8, E1 test cases	2019/04/08 20:05-20:20	Kellie Banzon
		Adding additional test cases, updating .in, .out, .expect files	2019/04/09 12:00-13:30	Kellie Banzon
Documentation	1.83	Skeleton, organization, task log	2019/04/07 22:30-23:00	Kellie Banzon
		Internal Architecture	2019/04/08 10:15-10:30	Cole Bemis, Kellie Banzon

		README	2019/04/11 15:10-15:40	Tanner Larson
		Testing	2019/04/10 10:20-10:30	Cole Bemis, Kellie Banzon
		Final Notes, edits & polishing	2019/04/10 10:20-10:45	Kellie Banzon

Testing *(when, who, how long, how many bugs found, how long it took to fix them)*

We created two scripts to aid in the testing process. The first, called “gen-expect.sh,” runs our program with all of the hand-written test input files (.in) and writes the output to .expect files. This convenience script made writing the expected output easier. We manually checked the generated output files and used these files as reference to diff against later. The second script, called “diff.sh,” runs our program with all the test input files (.in) and diffs the output (.out) with the expected output (.expect).

Tanner and Kellie each wrote and ran test cases (see the Task Log for exact time spent). We found a bug in the parsing that assumed required second words would be present and a mistake in the output of certain queries that did not print according to the spec. Since both of these bugs are quite simple errors, each only took a few minutes to fix.

Final Notes *(anything else you want to share about your implementation)*

The program supports lowercase command words (i.e. “bus” is valid input). However, student names must be in all caps (i.e. “student: cookus” will fail).