Lab 1 Write Up
Team Members: Flo Costa, Daniel Heyde, Steven Sun

Implementation Notes
We chose to use Python as the language for our implementation, as it is a relatively simple language that we all have experience with. For our architecture, we decided to store the data in several dictionaries, so that we could quickly index using a search term. We created 7 dictionaries with the following keys: student last name, teacher last name, bus number, grade (student), class number (student), class number (teacher), and grade (teacher). Each of these dictionaries contains a list student objects that correspond to that key. This approach allowed us to quickly have a student entry ready for any allowed query, without searching through any collections. While this approach allowed for fast querying and data recollection, it came at the cost of storage space. Since we stored every student and teacher in multiple locations, we used more space than we could have, had we used a slower implementation. Since our data set is small, we decided that using multiple dictionaries was a feasible option, however, if the data set was several orders of magnitude larger, we would change our approach.

Task Log
Flo:
Teacher Object:
- Start: 10:00am on Wed 4/10
- End: 11:00am on Wed 4/10
- Person Hours: 1

Updating and Adding Commands
- Start: 10:00am on Fri 4/12
- End: 11:00am on Fri 4/12
- Person Hours: 1

Test Files
- Start: 1:00pm on Sun 4/14
- End 2:00pm on Sun 4/14
- Person Hours: 1

Daniel:
Text file parsing and dictionary creation:
- Start: 10:00am on Wed 4/10
- End: 11:30 on Wed 4/10
- Person Hours: 1.5
- Start: 10:00am on Fri 4/12
- End: 10:30am on Fri 4/12
- Person Hours: 0.5

Steven:
Updating schoolsearch's parsing commands

- Start: 10:30am on Wed 4/10
- End: 11:00am on Wed 4/10
- Person Hours: 0.5
- Start: 10:00am on Fri 4/12
- End: 11:00am on Fri 4/12
- Person Hours: 1
- Start: 4:00pm on Fri 4/12
- End: 4:30am on Fri 4/12
- Person Hours: 0.5

Testing Notes

Flo wrote the test files and handled most of the testing. It took roughly 1 hour to write test and fix bugs.

- Bugs that were fixed:
    - Selecting the correct dictionary within each command
    - Incorrect arguments passed into commands
    - Changed how commands handled different options
    - Small typos

Modifications from Part 1 code:

- Used same structure that we had originally used in Part 1
- Added 3 new dictionaries which allowed us to further query data
- Implemented new commands to meet the new requirements
- Changed how some commands handled options to refine searches
- Added a new Teachers class which created teacher objects from the teacher file
- Modified dictionaries, teacher, and student objects to handle new files

Commands added:

- G[rade] <Number> T[eacher]
    - Given a grade number, provide all teacher who teaches that grade
- C[lassroom] <Number> S[tudent]
    - Given a classroom number, search for students assigned to it
- C[lassroom] <Number> T[eacher]
    - Given a classroom number, search for teachers teaches in it
- E[nrollment]
    - Gives the enrollment counts for each classroom ordered by classroom number
- D[ata] G[rade]

    - This command lists the GPA's of each student within each grade and calculates the average GPA for each grade
- D[ata] B[us]
    - This command lists the GPA's of each student for each bus route number and calculates the average GPA for each bus number
- D[ata] T[eacher]
    - This command lists the GPA's of each student for each teacher and calculates the average GPA for each teacher