CPE 365 Lab 1-2
Cole Grigsby
Avinash Sharma
4/8/17

Initial Decisions:

We decided to use Python 2 to implement this lab. This way, we can easily write methods for each functionality and run it easily from a unix command line.

Internal Architecture:

For part 2, the data was split up into two files: teachers.txt and list.txt in order to save space and avoid repeating the same data (teacher's names) in the students.txt file. Because of this we had to modify our Student object because it assumed the file of students would have that information, but list.txt did not contain that data. In addition, we needed to read in two files instead of one and we also decided to create a Teacher object to store the data from teachers.txt. Therefore, instead of having one list of Student objects, we had two lists: one of the new Student objects without the teacher's names and one of the Teacher objects, with each object representing a single line in the respective files.

Tasks:

- Read in files and parse into array of student and teacher objects - Avinash Sharma
    - Started and finished April 5th in lab (30 min)
- Each new interactive command - Cole Grigsby, Avinash Sharma
    - Met on Thursday and Friday in library to work on it (3.5 hours)
- Testing - Cole Grigsby, Avinash Sharma
    - While writing commands we wrote the test cases for them (1 hour during command writing, 15 mins to run all and store tests.out)
- Writeup - Cole Grigsby, Avinash Sharma
    - Started and finished April 8th (25 min)

Testing:

We wrote test cases as we designed methods for each command to test their functionality. After completing the commands, we tested everything together. We found a few small formatting errors that we were able to fix quickly such as, parsing our input using split by whitespace instead of split by a single space. We also fixed a small problem with a new line at the end of each line of the file by using the rstrip function. Some of the bugs had to do with maintaining two lists of two different objects (student list and teacher list), but it only took about 10 minutes to fix it. Once we got used to maintaining two lists, we didn't run into any bugs.

Final Notes:

We wrote a function to get the results for each command from the user input and printed out the results of those functions within our runProgram function.

Part 2

Decisions:

For part 2, the data was split up into two files: teachers.txt and list.txt in order to save space and avoid repeating the same data (teacher's names) in the students.txt file. Because of this we had to modify our Student object because it assumed the file of students would have that information, but list.txt did not contain that data. In addition, we needed to read in two files instead of one and we also decided to create a Teacher object to store the data from teachers.txt. Therefore, instead of having one list of Student objects, we had two lists: one of the new Student objects without the teacher's names and one of the Teacher objects. We also had to modify several of our methods that relied on having the teacher's first and last name within the Student object including, our studentLastName, teacherLastName, gradeTop, gradeLow, gradeTeachers, and teacherAverages functions. We also had to modify the way we stored and printed the results because we had to keep track of two lists.

Issues: We ran into an interesting issue with newlines for windows '/r/n' being a problem running on unix. It took a while to figure out with print statements were not working correctly but fixed by simply creating the list.txt and teachers.txt on a mac instead of windows.

Syntax and Semantics:

- **Requirement NR1** - Given a classroom number, list all students assigned to it
  Command : C[lass]: (input number) S[tudent]

Examples:
C: 101 S
Expected Output:
// COOKUS,XUAN
// ELHADDAD,SHANTE
// SWEDLUND,SHARRI
// CIGANEK,MANIE
// COVINGTON,TOMAS
// EARLY,TORY
// LINHART,LELA

Class: 108 Student
Expected Output:
// WORBINGTON,DEEDRA
// LIBRANDI,TODD
// HAVIR,BOBBIE
// SARAO,DIEDRA

// VANCOTT,MIKE
// WICINSKY,TERESE
// KOZOLA,BUSTER
// MULLINGS,LEIGHANN
// BUSSMANN,BILLY
// BERBES,DICK
// MULGREW,RANDELL
// TOWLEY,LANE

- **Requirement NR2** - Given a classroom number, find the teacher(s) teaching in it
Command : C[lass]: (input number) T[eacher]

Examples:
C: 112 T
Expected output:
// CHIONCHIO,PERLA

Class: 105 T
Expected output:
// HANTZ,JED

- **Requirement NR3** - Given a grade, find all teachers who teach it
Command : G[rade]: (input number) T[eacher]

Examples:
G: 4 T
Expected output:
// COOL,REUBEN
// HANTZ,JED
// CHIONCHIO,PERLA

Grade: 2 Teacher
Expected output:
// STEIB,GALE
// HAMER,GAVIN

- **Requirement NR4** - Report the enrollments broken down by classroom (output list of classrooms ordered by classroom number, with a total number of students per classroom)
Command : E[nrollment]

Examples:
E
Expected output:
// 101: 1
// 102: 5
// 103: 2

// 104: 2
// 105: 6
// 106: 2
// 107: 7
// 108: 11
// 109: 5
// 110: 2
// 111: 9
// 112: 8

Enrollment
Expected output:
// 101: 1
// 102: 5
// 103: 2
// 104: 2
// 105: 6
// 106: 2
// 107: 7
// 108: 11
// 109: 5
// 110: 2
// 111: 9
// 112: 8

- **Requirement NR5** - Display the average GPA and number of students per grade level
Command : GA or GradeAverage

Examples:
GA
Expected output:
// Grade: 0: Average GPA: 0, Number of Students: 0
// Grade: 1: Average GPA: 2.995, Number of Students: 2
// Grade: 2: Average GPA: 2.94615384615, Number of Students: 13
// Grade: 3: Average GPA: 3.04888888889, Number of Students: 9
// Grade: 4: Average GPA: 2.95133333333, Number of Students: 15
// Grade: 5: Average GPA: 0, Number of Students: 0
// Grade: 6: Average GPA: 2.97714285714, Number of Students: 21

GradeAverage
Expected output:
// Grade: 0: Average GPA: 0, Number of Students: 0
// Grade: 1: Average GPA: 2.995, Number of Students: 2
// Grade: 2: Average GPA: 2.94615384615, Number of Students: 13

// Grade: 3: Average GPA: 3.04888888889, Number of Students: 9
// Grade: 4: Average GPA: 2.95133333333, Number of Students: 15
// Grade: 5: Average GPA: 0, Number of Students: 0
// Grade: 6: Average GPA: 2.97714285714, Number of Students: 21

- **Requirement NR5** - Display the average GPA and number of students per teacher
Command : TA or TeacherAverage

Examples:

TA

Expected output:

// Teacher: ALPERT: Average GPA: 3.17, Number of Students: 2
// Teacher: BODZIONY: Average GPA: 3.09, Number of Students: 2
// Teacher: CHIONCHIO: Average GPA: 2.985, Number of Students: 8
// Teacher: COOL: Average GPA: 2.91, Number of Students: 1
// Teacher: FAFARD: Average GPA: 3.01428571429, Number of Students: 7
// Teacher: FALKER: Average GPA: 2.995, Number of Students: 2
// Teacher: GAMBREL: Average GPA: 2.96, Number of Students: 5
// Teacher: HAMER: Average GPA: 2.95454545455, Number of Students: 11
// Teacher: HANTZ: Average GPA: 2.91333333333, Number of Students: 6
// Teacher: KERBS: Average GPA: 2.976, Number of Students: 5
// Teacher: NISTENDIRK: Average GPA: 2.96222222222, Number of Students: 9
// Teacher: STEIB: Average GPA: 2.9, Number of Students: 2

TeacherAverage

Expected output:

// Teacher: ALPERT: Average GPA: 3.17, Number of Students: 2
// Teacher: BODZIONY: Average GPA: 3.09, Number of Students: 2
// Teacher: CHIONCHIO: Average GPA: 2.985, Number of Students: 8
// Teacher: COOL: Average GPA: 2.91, Number of Students: 1
// Teacher: FAFARD: Average GPA: 3.01428571429, Number of Students: 7
// Teacher: FALKER: Average GPA: 2.995, Number of Students: 2
// Teacher: GAMBREL: Average GPA: 2.96, Number of Students: 5
// Teacher: HAMER: Average GPA: 2.95454545455, Number of Students: 11
// Teacher: HANTZ: Average GPA: 2.91333333333, Number of Students: 6
// Teacher: KERBS: Average GPA: 2.976, Number of Students: 5
// Teacher: NISTENDIRK: Average GPA: 2.96222222222, Number of Students: 9
// Teacher: STEIB: Average GPA: 2.9, Number of Students: 2

- **Requirement NR5** - Display the average GPA and number of students per bus
Command : BA or BusAverage

Examples:

BA

Expected output:

// Bus Number: 0: Average GPA: 2.9525, Number of Students: 8
// Bus Number: 51: Average GPA: 3.02, Number of Students: 8
// Bus Number: 52: Average GPA: 2.885, Number of Students: 8
// Bus Number: 53: Average GPA: 3.05555555556, Number of Students: 9
// Bus Number: 54: Average GPA: 2.94166666667, Number of Students: 12
// Bus Number: 55: Average GPA: 3.03666666667, Number of Students: 9
// Bus Number: 56: Average GPA: 2.92166666667, Number of Students: 6

BusAverage

Expected output:

// Bus Number: 0: Average GPA: 2.9525, Number of Students: 8
// Bus Number: 51: Average GPA: 3.02, Number of Students: 8
// Bus Number: 52: Average GPA: 2.885, Number of Students: 8
// Bus Number: 53: Average GPA: 3.05555555556, Number of Students: 9
// Bus Number: 54: Average GPA: 2.94166666667, Number of Students: 12
// Bus Number: 55: Average GPA: 3.03666666667, Number of Students: 9
// Bus Number: 56: Average GPA: 2.92166666667, Number of Students: 6