CS 2316

Individual Homework 4 – Credit Standing

Due: Wednesday February 5th, 2013

Out of 100 points

Files to submit: **1. HW4.py**

This is an INDIVIDUAL assignment!

Collaboration at a reasonable level will not result in substantially similar code. Students may only collaborate with fellow students currently taking CS 2316, the TA's and the lecturer. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for others.

For Help:
- TA Helpdesk – Schedule posted on class website.
- Email TA's or use T-Square Forums

-Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).
-Do not wait until the last minute to do this assignment in case you run into problems.

Premise

In this assignment, you will be tasked with reading in data about a student's credits earned and find their current credit standing. Once you read in the data from a CSV file, you will be required to insert the data into a specified data structure which uses a Dictionary, lists, and tuples. After inserting all of the data into the data structure, you will be required to print out the structure in a human readable format and also write out a CSV file with some results. Details about the file you will be reading, the data structure you will be creating, and the output format will be covered later in this specification. In this assignment, you will be required to write the following five functions:

1. generateList()
2. parseLine()
3. readCSVFile()
4. writeCSVData()
5. printData()

File Format Information

The file will contain the following information: the GTID#, Last Name, First Name, GPA, followed by a series of numbers representing the number of credits taken each semester.
For Example:

902000001, Burdell, George, 4.0, 17,18,19, 21

The GPA will always have a decimal and the credits will always be integers. Each line will have an absolute minimum of one semesters worth of credit (one integer), and will likely have many semesters worth of credits. In the example above George Burdell has a 4.0 GPA and has taken 4 semesters of classes, with 17,18,19, and 21 credits respectively.

Data Structure Information

The primary data structure for this assignment will be a dictionary named **classification**. Remember that dictionaries contain a key, which is how you look up information in the dictionary, and each key has an associated value with it. In this particular dictionary, the key will be the GTID# of the student and the value will be a list, which will be constructed to look like this:

[(Last name, First name), GPA, Credit Total, Credit Classification]

Note that the GPA is a float, the Credit Total is an integer, while the Credit Classification, Last name and First name fields are strings.

Finding Credit Classification:

The Credit Total will be calculated as the sum of all the semester credits the student has taken and then the Credit Total will be used to find the Credit Standing of the students as indicated in the following table:

Freshman	0-29 credits
Sophomore	30-59 credits
Junior	60-89 credits
Senior	90-119 credits
Eligible to Graduate	120+ credits

Function Name: **generateList**

Parameters:

string – A string which contains the first name
string – A string which contains the last name
float – A floating point number representing the GPA
list – A list of integer credits

Return Value:

list – This list will have the value to be associated with the key in the dictionary in the required format

Description:

Write a function that will take the first name, the last name, the GPA and a list of all the credits that the student has taken so far. It will then calculate the total number of credits and the credit standing of the student based on the table included earlier. It will return a list in the correct format to insert into the dictionary as specified above following the format:

[(Last name, First name), GPA, Credit Total, Credit Classification]

Function Name: **parseLine**

Parameters:

string – A string which contains the contents of one line from the file
Dictionary – the data structure which to add the total credit and credit standing to.

Return Value:

Dictionary – the data structure containing the total credit and credit standing.

Description:

Write a function that will accept two parameters. The first is a string which contains the contents of one line from the file you are reading in the function readCSVFile(). The second is the dictionary to which you will be adding total credit and credit standing from the line to. The parseLine() function will then separate out the parts of the file. It should convert the GPA to a float and create a list of integers from all the credits that the student has taken. Then make the appropriate call to generateList() to generate the list that will be associated with that key in the dictionary. Finally, put the returned list into the dictionary data structure and return the dictionary.

Function Name: **readCSVFile**

Parameters:

string – A string which contains the name of the file to read in

Dictionary – An (likely empty) dictionary to add the data to.

Return Value:

Dictionary – the data structure containing the data.

Description:

Write a function that will accept two parameters. The first is a a string which contains the name of the CSV file you wish to parse (for example, “credit.csv”). The second is the dictionary into which you will add all of your data from the file. (It is likely that this dictionary will be empty when given to your function.) Your function will then open the file and read it's contents. Call parseLine() on each line to parse the information contained within that line. You must call parseLine() inside of readCSVFile() to parse each line of the file; the only thing that this function should do is read in the contents of the file and pass each line to parseLine(). You may assume that the file name will be valid. Remember to close the file when you’re done reading in the information. You ***may not use*** the CSV reader module; you will separate the data items yourself in the parseLine function described above. (Future homeworks will make use of the CSV module).
Note that your data file may have blank lines!

Function Name: **printData**

Parameters:

Dictionary – The data structure that has all the required data in the correct form

Return Value:

none

Description:

This function will ask the user to input a GTID#. The function will then display the information for the respective GTID# from your data structure in the following format:

GTID# FirstName LastName GPA CreditStanding
(Example: 902900001 George Burdell 4.0 Sophomore)

If the GTID# does not exist show the message “The GTID# entered does not exist” and return. (Do NOT ask for another GTID#).

Function Name: **writeCSVData**

Parameters:
Dictionary – the data structure that contains the ranks and the employee information.
Return Value:
None

Description:

Write a function that will accept one parameter that is a dictionary with key as GTID of each employee. The dictionary would have the following format:

Dictionary[GTID#]=[(Last name, First name), GPA, Credit Total, Credit Classification]

Use this information to write all the records of the students who are eligible to graduate from this dictionary into a CSV file called "eligibleGraduate.csv" in the following format:

GTID#, Last Name, First name, GPA, Credit Total

Remember to close the file when you're done writing the information.

Provided CODE:

Please include the following code (which uses all of your functions) in your file and make sure that your functions work correctly when used.

```
def creditClassification():
    aDict=readCSVFile("data.csv",  {} )
    writeCSVData(aDict)
    printData(aDict)



creditClassification()
```

Grading:

You will earn points as follows for each function that works correctly according to the specifications.

generateList()		**20**
Finds the correct credit standing	5	
Properly handles the name in a tuple	5	
Returns the list in the correct format	10	
parseLine()		**25**
Properly handles the GTID#,GPA and name	7	
Properly handles all the credits in the line	10	
Properly calls generateList()	5	
Correctly inserts data list into the data structure	3	
readCSVFile()		**12**
Properly handles multi-line files	5	
Properly ignores whitespace lines	3	
Properly passes lines in file to parseLine()	2	
Closes the file before exiting the function	1	
printData()		**23**
Properly handles input	5	
Properly handles input if GTID# does not exist	8	
Output contains all data in correct format	10	
writeCSVData()		**20**
Properly opens the file to write	5	
Uses the correct filename	2	
Properly writes the data in the correct format	8	
Closes the file before exiting	5	