(100 points) Programming Assignment 2 – Database Processing (Group Programming – up to 3 people in a group)

Read the entire document thoroughly!

1. Grading

--- 70% of the grade depends on your program's correctness. Brute force solutions and extremely inefficient programs that produce correct results may lose credit.
--- 30% of the grade depends on your program's readability. Programs should conform to the programming style emphasized throughout the semester. Meaningful variable names and a 'docstring' for the main program and every function are REQUIRED.

2. Objectives

_ First develop a general outline of how your menu-driven program will be structured. This outline is often called pseudocode showing the list of steps to take for solving the given problem. The outline is not in Python. In this process, you should identify the functions that are to be used, and any parameters that go with each function. (While you do not have to turn in your pseudocode, the first thing the TA or I will ask for if you come for help will be your pseudocode.)
_ Develop a Python program based on the designed pseudocode.
_ Test the program EXTENSIVELY for semantic (i.e., logic) errors. When appropriate, use assertEquals to test your functions.

4. Assignment

4.1 Overview

A database is a structured set of data held in a computer that can be easily accessed, managed and updated. This assignment involves creating an overly simplified database comprised of tables containing song data and providing a user with an interface to that database.

This database is comprised of 4 tables – songs, artists, top10, sales. The songs table contains the records of songs, and the artists table contains the records of artists, and so on. Each record contains fields. For example, the artists table contains records of artists, and each artist record contains the stage name field, the artist's full name field, and the artist's hometown field.

Download the following files from sakai program 2 folder for you to use for this program: artists.txt, sales.txt, songs.txt, top10.txt.

4.2 Program Design

Write a program in a file named *program2.py*.

Your program design should be "menu-driven" and should make extensive use of functions. Here is an outline of what the main program should do.

1. Display a welcome message and general instructions to the user. (Hint: call a function that outputs the welcome and instructions.)

2. Prompt the user with a menu for input:

s: load songs
a: load artists
t: load top 10
l: load sales
p: print a table – songs, artists, top10
q: query database – songs, artists
c: sales chart
w: write database to a file
e: exit the program

Verify the user enters a valid menu choice. If not, gracefully request the user to reenter a valid menu option. (Note: when an assignment says "Verify …", that implies your program must perform testing, providing a meaningful error message and, if appropriate, prompting the user to repeat his/her action.)
You must use a while loop. The option 'e' will exit the loop.

4.3 (10 pts for each function)
Write the functions for the operations. These are the minimum functions you must write.
Except the "exit the program" option, you must write a function for each operation.
The operations are defined as follows:

s: load songs
The function name should be loadSongs with no parameter. -3 pts if a different name is used. Prompt the user for the name of the file that has the song records. This file will be used to populate the songs 'table' in the 'database'. Read in the file into memory. Assume the filename is relative (i.e., the file is assumed to be located in the same folder as your program.) Assume each record (each line) of the file has the following format: 'songTitle|artistStageName\n' ('|', commonly called as a pipe, separates the fields) with types: string, string, respectively. Assume that there are no duplicate song titles. (Note: when an assignment says "Assume …", that means your program does NOT have to verify that assumption. It's ok if your program crashes, or behaves incorrectly if the assumption is untrue.)

Read the data from the file and load into a dictionary with the song title as the key and the song record as a list as the value. The list elements should be in the same order the fields are in each line of the file. See section 4.4 to create this dictionary.

For example, if the file contains lines of
Glory|Lotus\n
Hunger|Rose\n
The artists dictionary should contain
{'Glory':['Glory', 'Lotus'], 'Hunger':['Hunger', 'Rose']}

This function should return the loaded songs dictionary.

So, basically the structure of the songs table is a dictionary, and the song title (which is the key to the dictionary) is the "primary" key to the songs record, and the value is the song record that belongs to the song title. "Primary key" in database terms means it is a unique key ("identifier") to each record. This table uses a dictionary which means it is an "indexed" table (with primary keys) for quick searching by the song title (the primary key).

You can assume the user will enter a valid file name.

a: load artists

The function name should be loadArtists with no parameter. -3 pts if a different name is used. Prompt the user for the name of the file that has the artists records to be loaded into the artists table in the 'database'. Read in the file into memory. Assume the filename is relative. Assume each record of the file has the format: 'artistStageName|artistName|artistHomeTown\n' with types: string, string, string, respectively. Assume that there are no duplicate artist stage names.

Load the data read from the file into a dictionary with the artist stage name as the key and a list containing the artist data as the value. The artist data should be in the order of artist stage name, artist name, artist's hometown.
For example, if the file contains the following lines:
Lotus|Janet Kim|Newark, DE\n
Rose|Sharon Smith|Seattle, WA\n
The artists dictionary should contain
{'Lotus':['Lotus', 'Janet Kim', 'Newark, DE'], 'Rose':['Rose', 'Sharon Smith', 'Seattle, WA']}

This function returns the loaded artists dictionary. See section 4.4 to create this dictionary.

You can assume the user will enter a valid file name.

t: load top 10

The function name should be loadTop10 with no parameter. -3 pts if a different name is used. Prompt the user for the name of the file that has the top 10 records to be loaded into the top10 table in the 'database'. Read in the file into memory. Assume the filename is relative. Assume each record (i.e., each line) of the file has the format: 'songTitle\n'. Assume that there are no duplicate song titles.

Load the data read from the file into a list. The songs are in the order of top 1 to 10; therefore, the elements must be in the order of the rank.

The top10 table is a list that contains the top 10 songs.

This function returns the loaded top10 list.

You can assume the user will enter a valid file name.

l: load sales

The function name should be loadSales with no parameter. -3 pts if a different name is used. Prompt the user for the name of the file that has the records of the sales of the songs to be loaded into the sales table in the 'database'. Read in the file into memory. Assume the filename is relative. Assume each of the remaining records of the file has the following format: 'songTitle,amount,amount,amount,amount\n' with types: string, int, int, int, int, respectively. Assume that there are no duplicate song titles.

Then load the remaining data read from the file into a dictionary with the song title as the key and a list containing the sales data as the value. The sales data list should be in the same order the fields are listed in each record.

For example, if the file contains lines of
Hunger,234,324,453,234\n
Cheers,342,789,7866,9342\n
The sales dictionary should contain
{'Hunger':['Hunger', '234', '324', '453', '234'], 'Cheers':['Cheers', '342', '789', '7866', '9342']}

Note that these records are separated by commas.

The dates for the sales amount are 8/27/2015, 9/3/2015, 9/10/2015, 9/17/2015, respectively.

This function returns the loaded sales dictionary. See section 4.4 to create this dictionary.

You can assume the user will enter a valid file name.

p: print a table – valid tables are songs, artists, top10
The function name should be printATable with no parameter. -3 pts if a different name is used. Ask the user to enter a table name to print. The choices are songs, artists, or top10. Verify that user enters a correct table name. Print the entire table one record per line. If the table is a dictionary, do not print the dictionary key; you need to print only the values. The first line of the print out should be the field names.
For example, if the user wants to print the artists table, your print out should look like:

artistStageName	artistName	artistHomeTown
Crazy Boys	Jane Smith	Middle of Nowhere, AK
Python pupils	Tom Dickens	Center city, MI

For top10, list the top 10 list with the rank.
For example,

Rank	songTitle
1	Glory
2	Fallen

For this function, you must print the records using **recursion**. No points will be given if recursion is not used.

q: query database – valid tables are songs, artists
The function name should be query with no parameter. -3 pts if a different name is used.
Prompt the user to enter a query. The query should be in the following form:

select * from *tableName* where *primaryKey='value'*

- It translates to "display all fields of the record that has the primary key of *value* from table, *tableName*.
- Note that the value is in single quotes.

Your query function should parse the query string to pull the table name and the value, and print the record that belongs to the user specified table and the primary key value. You can assume

the user will enter the correct primary key for each table he/she is querying, so you do not need to validate primary key.

A query example is

if a user wants to query the songs table for the song that has the title of 'Glory', the query should look like:

```
select * from songs where songTitle='Glory'
```

and your print out should look like:

```
songTitle       artistStageName
-------------   ------------------------
Glory           Crazy Boys
```

You can assume the user will enter a valid query.

c: sales chart

The function name should be salesChart with no parameter. -3 pts if a different name is used.

a) Read the lecture slides and http://matplotlib.sourceforge.net/ about plotting in Python
b) The function should call the Python plotting software to plot sales amounts vs sales dates.
c) There should be multiple graphs on one figure. Each graph should represent each song sales.
d) Add legends indicating each graph.

See salesChart.png in the program 2 folder on sakai for an example.

w: write database to a file

The function name should be writeToFile with no parameter. -3 pts if a different name is used.
Write to a file named, musicLibrary.txt, the contents of the database in the following format:
songTitle|top10Rank|artistStageName|artistName|artistHomeTown|latestSalesAmount
where

songTitle is the song title
top10Rank is the top 10 rank
artistStageName is the song artist's stage name
artistHomeTown is the song artist's home town
latestSalesAmount is the last sales amount in the sales record – the last element in the sales record

See musicLibrary.txt in the program 2 folder on sakai for an example.

e: exit

Print a friendly goodbye message, and terminate the program.

4.4 Another function you must write (10 pts):

Write a function called createDictionary. The function should have at least one string parameter, filename. You will need to decide on any additional parameters you need for the createDictionary function. This function should read the contents of a file and then create and return a dictionary. The functions, loadSongs(), loadArtists(), and loadSales(), must call this function to create the dictionary they are required to return. There must not be any codes that create a dictionary in loadSongs(), loadArtists(), and loadSales() functions.

Hint: the only difference in the structure of the songs, artists, and the sales file lines/records is the character (commonly called a delimiter) that separates the fields.

4.5. Main program (10 pts):
Using the functions in sections 4.3 and 4.4 and any additional functions you need to complete this program, write your database program!

4.6 Sample Run

See program2shellscript.py in the program 2 folder on sakai.

5. Submit 4 files to Sakai:

After all the programmers in your group are ABSOLUTELY sure your program works, start a new Python shell with IDLE, and open your *program2.py.* Run your program exercising all of its functionality. Demonstrate fully how your program handles both valid and invalid input values. (Your program need not handle invalid input *types*.) After fully exercising your program, save your sales chart in a file named salesChart.png and your shell script session in a file named *program2shellscript.py*

Electronically upload 4 files – program2.py, program2shellscript.py, salesChart.png, and musicLibrary.txt. You must use the indicated filenames exactly (-5pts otherwise).

Be sure to include all students' names in the comments of *program2.py.*