CMPS 1500 Introduction to Computer Science I – Spring 2019

Lab 4

Due **Wednesday 2/20/18** at 11:59 p.m.

Start early and give yourself enough time to work on this lab.

Note that code formatting requirements will apply to this lab. Part of the lab grade will be for quality of formatting of the code. Parts of the lab will be graded by humans. The docstring for every function you write in this lab should contain a brief explanation of the input and the output of the function, including their types. You will be submitting problem 0 to Zybook, and problems 1 and 2 to Canvas.

In this lab, you will create a program that simulates the work of an interactive data management system. For example *Gibson*, the sysstem that is used to manage student information at Tulane, likely grew from a system very similar to the one that you will implement in this lab. The role of the *database* (collection of records) will be played by Python *dictionary*, and you will write functions that will manipulate data in the dictionary. In our example, we will keep a record of students and their associated majors, however, the same ideas will apply in any real-life applications where we need to keep and update records, such as a database of stocks and their prices, employees and their salaries at an HR office, accounts and their balances in a bank, and so on.

0. **Dictionary manipulation,** `lab4pr0.py`

   (This is a long problem and counts as double the points.)

   For this exercise, start by having a predefined dictionary storing a few student names and majors, such as:

```
majors = 'Harry':'Computer Science', 'Hermione':'Mathematics', 'Ron':'English'
```

   Place the line with your dictionary initialization at the beginning of your file.

   As you can see, the name will serve as a key, and the major will be the value. For simplicity, we'll assume that each person will only have one major.

   (a) Write a function `look_up(d)`, where `d` is a dictionary containing names and majors. The function should ask the user to enter a name. If the name is in the dictionary, it should print out the major associated with the name. If the name is not in the dictionary, it should print "Not found." The function should not return anything. Sample output of your function should look like this:

```
>>> look_up(majors)
Enter a name: Harry
Computer Science
>>> look_up(majors)
Enter a name: Ron
English
>>> look_up(majors)
Enter a name: Voldemort
Not found.
```

(b) Now write a function `add(d)` that asks the user to enter a name and a major. It then adds this information as a new entry in the dictionary `d`. If the name entered already exists in the dictionary, do not alter the dictionary; just print out "A person with this name already exists in the system." The function should not return anything. Sample output of your function should look like this:

```
>>> add(majors)
Enter a name: Hagrid
Enter a major: Physics
>>> majors
{'Harry': 'Computer Science', 'Ron': 'English', 'Hagrid': 'Physics',
'Hermione': 'Mathematics'}
>>> add(majors)
Enter a name: Ron
Enter a major: Art
A person with this name already exists in the system.
```

(c) Now write a function `change(d)` that allows a user to change an entry of the dictionary `d`. The function should ask the user to enter a name. If the name is not already in the dictionary, it should print, "That name is not found." If the name is in the dictionary, the function should ask the user to enter a major. Then the function should change the dictionary entry to include the new major. The function should not return anything. Sample output of your function would look like this:

```
>>> change(majors)
Enter a name: Harry
Enter the new major: Biology
>>> majors
{'Ron': 'English', 'Hermione': 'Mathematics', 'Harry': 'Biology'}
>>> change(majors)
Enter a name: Voldemort
That name is not found.
```

(d) Now write a function `delete(d)` that allows a user to delete an entry from the dictionary `d`. The function should ask the user for a name. If the name is in the dictionary, its entry should be removed. If the name is not in the dictionary, the function should print, "That name is not found." The function should not return anything. Sample output of you function should look like this:

```
>>> delete(majors)
Enter a name: Harry
>>> majors
{'Hermione': 'Mathematics', 'Ron': 'English'}
>>> delete(majors)
Enter a name: Voldemort
That name is not found.
```

(e) Write a function `display(d)` that displays all entries in a dictionary, one name/major pair per line, as follows:

```
>>> display(majors)
Hermione is a wizard in Mathematics
Harry is a wizard in Computer Science
Ron is a wizard in English
```

Note that a dictionary does not preserve the order of entries. Students that were added later may appear in the beginning of the list; don't be puzzled by that.

(f) Write a function `get_menu_choice()` that displays to the user all of the possible choices: look up, add, change, delete, and quit. Each choice will have a number associated with it. Then the function asks the user to make a choice. If the user enters an invalid choice, the function should ask again for a choice. Once a valid choice is entered, the choice should be returned. Sample output is as follows:

```
>>> get_menu_choice()

Majors of College Students
--------------------------
1. Look up a student's major
2. Add a new student
3. Change a major
4. Delete a student
5. Display all students
6. Quit the program

Enter your choice: 7
Enter a valid choice: hello
Enter a valid choice: 2
2
```

The main part of the program has been written for you. Open the lab on Zybook and look at the main code. It starts with an empty dictionary, `majors`. Now the `get_menu_choice` function is used to ask the user for a choice. When the user makes a choice, the corresponding function will be called to alter or access the `majors` dictionary. Then the user will be prompted to enter another choice. This process continues until the user enters the choice of quitting the program.

Develop the functions `look_up, add, change, delete, display, get_menu_choice` in a single file in IDLE, and test them thoroughly. Part of your test should be to copy the main code from Zybook at the bottom of your file and run it. A sample run is presented below.

For grading, copy all the functions that you have developed above the main code in the Zylab.

```
>>>

Majors of College Students
```

```
--------------------------
1. Look up a student's major
2. Add a new student
3. Change a major
4. Delete a student
5. Display all students
6. Quit the program

Enter your choice: 2
Enter a name: Hagrid
Enter a major: Computer Science

Majors of College Students
--------------------------
1. Look up a student's major
2. Add a new student
3. Change a major
4. Delete a student
5. Display all students
6. Quit the program

Enter your choice: 2
Enter a name: Hedwig
Enter a major: Mathematics

Majors of College Students
--------------------------
1. Look up a student's major
2. Add a new student
3. Change a major
4. Delete a student
5. Display all students
6. Quit the program

Enter your choice: 1
Enter a name: Hedwig
Mathematics

Majors of College Students
--------------------------
1. Look up a student's major
2. Add a new student
3. Change a major
4. Delete a student
5. Display all students
6. Quit the program
```

```
Enter your choice: 3
Enter a name: Hagrid
Enter the new major: Physics

Majors of College Students
--------------------------
1. Look up a student's major
2. Add a new student
3. Change a major
4. Delete a student
5. Display all students
6. Quit the program

Enter your choice: 1
Enter a name: Hagrid
Physics

Majors of College Students
--------------------------
1. Look up a student's major
2. Add a new student
3. Change a major
4. Delete a student
5. Display all students
6. Quit the program

Enter your choice: 4
Enter a name: Hedwig

Majors of College Students
--------------------------
1. Look up a student's major
2. Add a new student
3. Change a major
4. Delete a student
5. Display all students
6. Quit the program

Enter your choice: 1
Enter a name: Hedwig
Not found.

Majors of College Students
--------------------------
1. Look up a student's major
2. Add a new student
3. Change a major
```

```
4. Delete a student
5. Display all students
6. Quit the program

Enter your choice: 6
```

1. **Persistence**, `lab4pr1.py`

   Modify the your program so that it stores the dictionary in the file "dictionary.txt" between successive runs of your program. That is, in the very beginning of the main part of the program it should check whether the file "dictionary.txt" exists and if yes, it should read the data from the file into the dictionary. At the very end of the main program, it should write the dictionary content into "dictionary.txt". How you store the data into the file is really up to you. The reading and writing parts should be handled by separate functions, to avoid cluttering the main part of the program.

2. **Computer history exhibit**, `lab4pr2.txt`

   In this assignment you will take the role of a museum curator. You will select a piece to represent a milestone in computer technology development. Together with your class you will create a CMPS 1500 virtual exhibit of computer history. To start, watch a TED talk by John Graham-Cumming "The greatest machine that never was" (12 minutes)

   `https://www.ted.com/talks/john_graham_cumming_the_greatest_machine_that_never_was?language=en`

   about the machine that was the first computer and about the first programmer. Then take some time to visit the exhibits Computer History Museum

   `http://www.computerhistory.org/exhibits/`

   Select one piece for our virtual exhibit. It can be a computer program, a hardware device (e.g., the first mouse) or a computing device, it can be an idea or its realization, or a phenomenon, or an event. Essentially something that changed the path of computer technology development. It doesn't have to be in use today.

   For your answer, submit a link to your piece (an article or video about it), the year when the piece first appeared, and a 3-5 line description of why you believe this is a crucial piece for the exhibit, what was its role in computer history.

   This is not a knowledge-based assignment, you're not expected to know much computer history going into this class. You should spend time reading and researching before you provide the answer. Your link doesn't have to go to computerhistory.org, if you find an interesting video on youtube or elsewhere (or maybe you'd like to make one?), you should include it instead. Remember to provide the specific detail that identifies your piece (don't just say "a mouse", mention make, model and year for the mouse, and so on). (The examples used in the text of this assignment - Babbage engine and mouse - are already taken and cannot be used as an answer).