UNIVERSITY OF SASKATCHEWAN

Department of Computer Science
176 Thorvaldson Building
110 Science Place, Saskatoon, SK, S7N 5C9, Canada
Telephine: (306) 966-4886, Facimile: (306) 966-4884

Assignment 5

Dictionaries and File I/O

Date Due: Friday, October 25, 2019, 6:00pm **Total Marks: 23**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work.
- For this assignment, you will hand in one single file, called `a5.py`. Put your name and student number at the top of the document. This assists the markers in their work. Failure to follow these conventions will result in needless effort by the markers, and a deduction of grades for you. Do not submit folders, zip documents, even if you think it will help.
- Programs must be written in Python 3.
- **Assignments must be submitted to Moodle.** There is a link on the course webpage that shows you how to do this.
- **Moodle will not let you submit work after the assignment deadline.** It is advisable to hand in each answer that you are happy with as you go. You can always revise and resubmit as many times as you like before the deadline; only your most recent submission will be graded.
- Questions are annotated use descriptors like "easy" "moderate" and "tricky". All students should be able to obtain perfect grades on "easy" problems. Most students should obtain perfect grades on "moderate" problems. The problems marked "tricky" may require significantly more time, and only the top students should expect to get perfect grades on these. We use these annotations to help students be aware of the differences, and also to help students allocate their time wisely. Partial credit will be given as appropriate, so hand in all attempts.

Overview

Cities are highly complex systems, yet we rarely think about them that way. They produce and have access to lots of **data**! The City of Saskatoon even has an Open Data Catalogue (*http://opendata-saskatoon.cloudapp.net/*) which anyone can access. In this assignment, we are going to use Python to build a program that parses some of this data, and analyzes it at the neighbourhood level.

Saskatoon uses as ward system in its civic election system. Meaning that Saskatoon is divided evenly into several **wards**. And each ward is divided into several **neighbourhoods**. In this assignemnt, we will be looking specifically at data that tells us which languages are spoken in each neighbourhood.

This assignment is divided into 5 different "questions", but in the end, your work from all of the questions will be combined to complete the overall program. The first four questions will each ask you to define a function that will perform a specific task. Make sure that the inputs (parameters) and outputs (return values) of your functions match the problem description **exactly**, or else the whole thing won't fit together properly!

The following is a brief overview of how the questions for this assignment will fit together:

- **Question 1**: Read data about the city from a text file and organize it as a **database**.
  You must fully complete this question before moving on.
  All of the other questions will depend on the database that you create here.
- **Question 2**: Build a list of all the **unique** *wards* in the city.
- **Question 3**: Build a list of all the *neighbourhoods* in a given ward.
- **Question 4**: Count how many times each *language* occurs within a given list of neighbourhoods.
- **Question 5**: For each *ward*, display the *neighbourhood language* counts.

Individually, none of these questions are particularly difficult, but if you make a mistake anywhere along the way, then the last part won't work. Make sure to perform simple testing, such as printing out lists or dictionaries, as you go to make sure that everything looks right.

One last note of caution: We have given you an input file for this assignment. Be aware that we will test your program with a different input file, that contains **different** neighbourhood data. Therefore, you must make sure your code is fully general; don't count on knowing exactly how many neighbourhoods/wards there are, or on knowing exactly what languages there might be.

What to Hand In

When you're done, you'll hand in one document, `a5.py`, with **ALL** your work in it. Make sure you put identification information at the top of your program!

Question 1 (6 points):

Purpose: To practice the concept of dictionary as a database

Degree of Difficulty: Moderate.

Before starting this question, first, download the data file `CityData.txt` from the class Moodle. Make sure to put it in the same folder as your `a5.py` python code.

Write a function called `read_citydata()` that takes as parameter(s):

- a string indicating the name of a city data file

This function should **return** a database (i.e. a dictionary-of-dictionaries) that stores all of the City data in a format that we will describe further below. You can review section 11.1.9 of the text for a refresher on what databases look like using dictionaries.

Input File Format

The data file for this question looks like this:

```
Holiday Park,Ward 2,Spanish,English,French,Ukrainian,Cree,German
North Park,Ward 1,Tagalog,Ukrainian,French,German,English
```

Each line of the file contains all of the data for a single neighbourhood. The first item on the line is always the neighbourhood's name; names are guaranteed to be unique. The second item is always the neighbourhood's ward, which in the example above, is `Ward 2`. Following that are **one or more** languages which are spoken in that neighbourhood. So there are residents in Holiday Park that speak Spanish, English, French, Ukrainian, Cree, or German. Note that the language names CAN contain spaces. All of the data items on a each line are separated by commas.

Database Format

Your function should open the file given as a parameter and read the data from it into a database. The keys for this database will be neighbourhood names, and the values will be records (i.e another dictionary) for the matching neighbourhood.

First, the `read_citydata()` function should create an empty dictionary to be the overall database.

Then, for each neighbourhood from the input file, create one record (i.e. one dictionary) with the following fields:

- The neighbourhood's name, as a string.
- The ward, as a string.
- The spoken languages, as a **list** of strings.

This record should then be added to the database, using the neighbourhood's name as a key.

Once all of the records have been added, the function should return the database dictionary.

Evaluation

- 1 mark: Function correctly uses file name parameter to open the data file
- 1 mark: The function returns a database dictionary.
- 2 marks: The record for each neighbourhood is correctly constructed

- 1 mark: Each record is correctly added to the database dictionary
- 1 mark: For a good function docstring

Question 2 (4 points):

Purpose: To practice iterating over data

Degree of Difficulty: Easy.

Write a function called `find_wards()` which takes as parameter(s):

- A dictionary in the format of the city database as described in Q1

This function should construct and **return** a **list** of all the different wards. Each entry in the list should be unique; in other words, don't add the same ward twice. For the input file you were given, the resulting list should be (though not necessarily in this order):

```
['Ward 1', 'Ward 9', 'Ward 5', 'Ward 2', 'Ward 3',
'Ward 4', 'Ward 6', 'Ward 8', 'Ward 7', 'Ward 10']
```

Also, note that this function **must not change** the city database in any way.

Evaluation

- 1 mark: The function parameters and return value are the correct data type
- 1 mark: All of the wards are in the returned list
- 1 mark: The list entries are all unique
- 1 mark: For a good function docstring

Question 3 (4 points):

Purpose: To practice iterating over nested compound data and accessing different dictionary fields

Degree of Difficulty: Easy.

Write a function called `ward_neighbourhoods()` which takes as parameter(s):

- A dictionary in the format of the city database as described in Q1
- The name of a ward, as a string

The function should construct and **return** a **list** containing the **neighbourhoods** that can be found in a particular ward. For example, for the input file you were given, if the ward is `"Ward 2"`, the resulting list should be (though not necessarily in this order):

```
['Meadowgreen', 'Westmount', 'Pleasant Hill', 'Caswell Hill',
'Montgomery Place', 'King George', 'Holiday Park', 'Riversdale']
```

Also, note that this function **must not change** the city database in any way.

Evaluation

- 1 mark: The function parameters and return value are the correct data type
- 2 marks: All of the neighbourhoods in the given ward are found and placed in the list
- 1 mark: For a good function docstring

Question 4 (4 points):

Purpose: To practice summarizing data from a dictionary

Degree of Difficulty: Moderate.

Write a function called `count_languages()` which takes as parameter(s):

- A dictionary in the format of the city database as described in Q1
- A list of neighbourhood names (as strings)

This function should construct and **return** a **dictionary** where each key is a **language** (e.g. `English` or `French`, etc...) and the value for that key is the number of neighbourhoods of that speak that language in a given list.

For example, using the sample list of all of the neighbourhoods from `"Ward 2"` (see Q3 to see that list), the resulting dictionary should be:

```
{'Sino-Tibetan languages': 2, 'Chinese': 1, 'Mandarin': 1, 'Arabic': 1,
'Bengali': 1, 'French': 5, 'English': 8, 'German': 4, 'Cree': 6, 'Tagalog': 4,
'Ukrainian': 5, 'Spanish': 2, 'Cantonese': 1}
```

Also, note that this function **must not change** the city database or the given list of neighbourhood names in any way.

Evaluation

- **-1 mark** for lack of identification in the solution.
- 1 mark: The function parameters and return value are the correct data type
- 1 mark: The format of the returned dictionary is correct
- 1 mark: The counts are correctly calculated
- 1 mark: For a good function docstring

Question 5 (5 points):

Purpose: To practice solving a problem by calling multiple functions; to practice passing data results between functions

Degree of Difficulty: Easy if you did everything right so far

Now it's time to put all your functions together to print out a summary report of the number of neighbourhoods that speak unique languages in each ward.

In the main part of your program, write code that will **print to the console** the name of each ward, the TOTAL number of different neighbourhoods that can be found in that ward, followed by the counts for **number of neighbourhoods** for each spoken language in that ward.

For example, the format for your output might look something like this:

```
Ward 1 contains 10 neighbourhoods.
Below are the languages spoken in the Ward
and the number of neighbourhoods where that language is spoken.
{'Russian': 1, 'Telugu': 1, 'Hindi': 1, 'Urdu': 2, 'Chinese': 1,
'Mandarin': 1, 'Arabic': 3, 'German': 8, 'French': 8, 'English': 10,
'Bengali': 1, 'Cree': 2, 'Tagalog': 5, 'Punjabi': 1, 'Gujarati': 1,
'Ukrainian': 6}

Ward 2 contains 8 neighbourhoods.
Below are the languages spoken in the Ward
and the number of neighbourhoods where that language is spoken.
{'Sino-Tibetan languages': 2, 'Chinese': 1, 'Mandarin': 1, 'Arabic': 1,
'Bengali': 1, 'French': 5, 'English': 8, 'German': 4, 'Cree': 6,
'Tagalog': 4, 'Ukrainian': 5, 'Spanish': 2, 'Cantonese': 1}
```

There will be more wards, but we're only showing two here to save space.

To do this, you will need to make use of all of the functions you have written so far and make good use of their inputs and outputs. If you've done everything correctly, this part of your program won't require a lot of code: just a few function calls and a **single for-loop** should be enough.

Evaluation

- **-1 mark** for lack of identification in the solution file.
- 4 marks: For making correct and useful function calls to each of the functions from Q1-Q4
- 1 mark: For correct output printed to the console.