# Programming Project 8

(Hints added on 11/3)

This assignment is worth 50 points (5.0% of the course grade) and must be completed and turned in before 11:59 on Monday, November 14, 2016.

**Assignment Overview**

**The Problem**

Computer-based translation is one of the most important and challenging problems facing computer science. Because of the significant differences between the semantics of different languages, approaching this problem involves much more than translating each word and concatenating the resulting translations. However, in this assignment we will take that simple approach: simply translate word by word. In this project you are supposed to translate a paragraph originally written in English to 7 other languages. We have selected some of the most widely used languages according to Wikipedia: https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers

**Objectives**

The main goal of this project is to help you understand the notion of dictionaries versus the notion of lists in Python (and the corresponding data structures in any other programming language). After completing your part of the code, you should be able to pinpoint performance differences between the two of them.

**Assignment Deliverable**

The deliverable for this assignment is the following file:

`proj08.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via the **handin system** before the project deadline.

**Background**

The Text Encoding Initiative (TEI) is a consortium which collectively develops and maintains a standard for the representation of texts in digital form. They have developed the TEI format which has become the standard method of storing texts in digital form. You can read more about TEI here: http://www.tei-c.org

One of the organizations using this format is http://www.freedict.org which offers an extensive and free set of bilingual dictionaries. For our project we will use 7 of their dictionaries to convert from English to Arabic, Spanish, Hindi, Russian, French, German and Italian! All their dictionaries are available for download at https://github.com/freedict/fd-dictionaries, but we have copied the 7 we need into the project directory. They are provided both individually and collected together into a zip file for your convenience. They are large.

**Project Description / Specification**

Overview: we provide a working program based on lists; you code a solution based on dictionaries.

Most of the code for this project has been already done for you, except for the most important part! Completing the missing code is your task. Our TEI parsing logic is encapsulated in tei_parser.py, feel free to take a look, but do not modify anything (you do not need to) as we will be using our own version of the file while grading. This code uses two forms of data representations (data structures) for its operation. The first representation is list-based. Each database (DB)—the term 'database' is used loosely in our context to refer to the data structure holding the words along with their translations—is represented using only lists. The code for this section is completely done for you. You will see that without modifying anything, that just running the file will display 7 translations for your original text. We claim however that this is not an efficient way of approaching this problem! At the end of your output you will see a 'Running-Time Summary' section. On our machine it took approximately 200 milliseconds to translate a small paragraph. We also claim that using another data representation can drastically decrease the time required for translation (do net be deceived by the 0.00 milliseconds displayed at the end of your output. We cannot take it down that far of course! This happens only because you have not done your part yet.).

Our second form of representation is a dictionary-based structure. This is your part. You need to convert the already existing list-based representation into a dictionary-based representation.

1. `getDictFormattedDatabases(listDatabases)`. Your first task is to implement this function by replacing the filler statement `pass` with Python code. The parameter is a list of tuples where the second item in the tuple is a list of tuples of the form (word, translation). That is, the parameter `listDatabases` is a list of tuples, structured as follows:

```
[(lang1-name, list-based-DB1), (lang2-name, list-based-DB2) ...]
```

For example:

```
[("Arabic", arabicListBasedDb), ("Hindi", hindiListBasedDb) ...]
```

A single list-based DB (the second element in each of the tuples above) takes again the form of a list of tuples, as follows:

```
[(word1, word1-translation), (word2, word2-translation) ...]
```

For example,

```
[('have','avoir'), ('baby','bébé') ...]
```

The function returns a list of tuples where the second item in the tuple is no longer a list, but has been converted to a dictionary of items in the form word:translation. That is, the following form:

```
[(lang1-name, dict-based-DB1), (lang2-name, dict-based-DB2) ...]
```

For example,

```
[("Arabic", arabicDictBasedDb), ("Hindi", hindiDictBasedDb) ...]
```

A single dict-based DB (the second element in each of the tuples above) takes the following form:

```
{word1:word1-translation, word2:word2-translation ...}
```

For example,

```
{'have':'avoir', 'baby':'bébé' ...}
```

2) `translateFromDict(text, dictBasedDb)` The logic of this method is supposed to return the translation of the first parameter (`text`, a string) to the target language defined by the second parameter (`dictBasedDb`). Think of the parameter `text` as a word to be translated. Simply do a word-to-word translation. Special care should be given to those words followed by a full stop (a.k.a .period). Notice that due to the initial preprocessing of the inputted text, a full stop will always be adjacent to its preceding word and at least one space away from its subsequent word. If a word that

has a period attached, its translation must have a period attached. Return a string that is a translation of the parameter `text`. If `text` is not in the `dictBasedDb`, simply return the `text`.

`3) performDictBasedTranslations(preprocessedText,dictBasedDatabases)`

Finally, complete the implementation of the function `performDictBasedTranslations`. The parameter `preprocessedText` is the complete text to be translated; `dictBasedDatabases` is a list of all the language translation databases. This function should loop over all your dict-based databases to translate the `preprocessedText` into each of the languages. Display the name of the language and the translation using provided `display` function.

3) You may use other functions—I didn't for this project.
4) In the original language DB there may be multiple translations for one English word. We ignore that so that any one of those translations is accepted—we don't care which.

**Hints**

As with most problems break this problem down into smaller pieces. Like the last project it is best to start with a smaller set of input data. In this case you can "easily" create it.

a. Begin by not using the provided `proj08.py`. That is, build a separate test program starting from scratch.
b. Create your own `listDatabases` as described in the project description above for `getDictFormattedDatabases`. The original "ListBasedDb" have 6K or so entries (tuples). For testing you only need two or three (see next item). Instead of 7 languages, only use two or three. Constructing this is an excellent example of something to do with others because this will not be part of your final project—it is only for testing. Note that each tuple has an English word, paired with that word translated into the language of choice, e.g. if Spanish `('hello', 'hola').` (note that the translated word could be gibberish, e.g. `('hello', 'xxxx').`)
c. Create your own `preprocessedText`. Use something really simple such as `'hello world.'` (note that I included a period). If you use 'hello world', then include those two words in your `listDatabases`.
d. Now you can write your own `getDictFormattedDatabases` function. It will take your `listDatabases` (the list-based objects) and return the same information with dictionaries (actually list of tuples containing dictionaries).
e. In a similar way you can create the other specified functions.
f. Once you get them all working with your small `listDatabases` you can copy your code into the provided `proj08.py` to complete and test the project.

**Sample Output**

There is no sample output. The proj08.py we provide generates all the translations using lists. Your program will generate the same output—that is, the same translation will be printed twice. At the end of the program some run times will be printed—different computers will generate different times.

==========

**Educational Research**

**When you have completed the project insert the 5-line comment specified below.**

For each of the following statements, please respond with how much they apply to your experience completing the programming project, on the following scale:

**1** = Strongly disagree / Not true of me at all
**2**
**3**
**4** = Neither agree nor disagree / Somewhat true of me
**5**
**6**
**7** = Strongly agree / Extremely true of me

****Please note that your responses to these questions will not affect your project grade, so please answer as honestly as possible.****

**Q1: Upon completing the project, I felt proud/accomplished**

**Q2: While working on the project, I often felt frustrated/annoyed**

**Q3: While working on the project, I felt inadequate/stupid**

**Q4: Considering the difficulty of this course, the teacher, and my skills, I think I will do well in this course.**

Please insert your answers into the bottom of your project program as a comment, formatted exactly as follows (so we can write a program to extract them).

```
# Questions
# Q1: 5
# Q2: 3
# Q3: 4
# Q4: 6
```