FIT2004 S1/2019: Assessment questions for week 6

THIS PRAC IS ASSESSED! (7.5 Marks)

DEADLINE: Sunday, 7th April 2019 23:55:00 AEST
LATE SUBMISSION PENALTY: 20% penalty per day. Submitting after 5 days of deadline, you will get 0. Even more, if you are late less than a day, you will also be penalized accordingly.
PROGRAMMING CRITERIA: It is required that you implement this exercise strictly using **Python programming language** (version should not be earlier than 3.5). This practical work will be marked on the time, space complexity and output of your program.
SUBMISSION REQUIREMENT: You will submit a zipped file (should be named as ***student id.zip***, e.g. if your student id is 123456789, the name of zipped file must be `123456789.zip`. Moreover, your zipped the file maybe in `rar` or `7z` extension) containing your Python program file (named `decipher.py`) as well as a PDF file (named `Report_2.pdf`) briefly describing your solution and its space and time complexity. The PDF file must give an outline of your solution (e.g., a high level idea of how did you solve it) and the **worst-case** space and time complexity of your solution. Penalties will be applied if you fail to submit the PDF file. The zipped file is to be submitted on Moodle before the deadline. Draft submission will be considered as not-submitted and will not be marked.
PLAGIARISM: The assignments will be checked for plagiarism using an advanced plagiarism detector. Last year, many students were detected by the plagiarism detector and almost all got zero mark for the assignment and, as a result, many failed the unit. "Helping" others is NOT ACCEPTED. Please do not share your solutions partially or/and completely to others. If someone asks you for help, ask them to visit us during consultation hours for help.

Text decipher problem

Amelia, a friend of Jonathan. She wants to build a program to decipher a message from two encrypted texts, but does not know how to develop this decipher task using python efficiently. On the other hand, Jonathan is super impressed to you on solving the `Text count helper` and has recommended Amelia to contact you. (Please read the conversation very carefully and retrieve the tasks.)
You: "Hi Amelia, How are you?"
Amelia: "Hi, I am good. Thank you".
You said "Jon talked about your text deciphering. Let's discuss on your decipher project. Please explain the task in details"
Amelia: "I have two texts which encrypt the original message with lots of inessential alphabets.". "For an example: `bdceMlonfashxzunivyeriasityb` and `MondcaitjshukniveptnQrsiXteyY` are the two encrypted texts. Inside these two texts, ***Monash university*** is the original message."- Amelia
You: "Well, you are looking for longest sequence of common alphabets. It's easy task. Consider it is done."

You: “But before drive into the code, I have some queries in the task.”
You: “In the example, no space exits in each encrypted text. Is there any whitespace characters exist in the encrypted texts?
Amelia: “No.”
You: “That means the initial deciphered message will not have any whitespace character between the alphabets. We need to find out the words of the deciphered text. A list of possible words will help us to put the spaces in deciphered text to get original message.”
“So, do you have any dictionary to separate these words by space?” You asked
Amelia: “Yes, I do.”
You: “Awesome. At first we will find out the largest subsequence of common alphabets from two encrypted texts and then we will break the retrieved message into words based on the vocabulary.”
You: “Ah, what will happen if the combination of two or more words of dictionary is another words in the dictionary and present in the message. For an example: message is IloveMango. In your dictionary an, Man, go and Mango are available. Are we going to use Mango instead of an, Man and go?” Amelia replied “Certainly, it will be Mango.”
You: “By the way, if your dictionary is empty, what will we do?”
Amelia: “In this situation, the deciphered message will be read as a single word.”
You: “Just in case, words inside the dictionary are not matching, are we going to consider the message as a single words?”
Amelia: “I think so.”
“Thanks. That’s enough for me. When do you need this program?”-You. “I believe, I need this before 07 April 2019.”-Amelia.
You: “Hmm, only 14 days! I will finish it before the deadline”. “Thanks” -Amelia. “Cheers”-You. Amelia-“Bye”. “Bye”-You.
You have only 14 days to prepare the solution to decipher two texts, so start working on it now.

Input

There are two input files required to complete this text decipher problem. One input file named `encrypted.txt` consists of two variable sized encrypted texts. Each text contains of only English alphabets (a,b,c,....z), in both lower and uppercase format. There is no punctuation or whitespace character in each text. In the input file, two texts will be in two consecutive lines, not in same line. Let consider n and m be the size of first text and second text respectively. Therefore, the input size of first file is $O(n + m)$.

```
bdceMlonfashxzunivyeriasityb
MondcaitjshukniveptnQrsiXteyY
```

Another input file will contain all possible set of words. Each word is the combination of only upper and (or) lower case alphabets, no punctuation or whitespace character. The name of this input file is `dictionary.txt`. Let’s assume that N be the number of words in `dictionary.txt` and M be the maximal size of the words in the file. Therefore, the input size of second file is $O(NM)$.

```
Monash
university
```

```
Melbourne
Ballarat
college
school
```

Your program must be able to do two tasks as described below, where the output of previous task is input to next task.

Task 1: Finding the longest subsequence of common alphabets

In this task, your program will decipher the message (finding out the longest subsequence of common alphabets) from two encrypted texts, input from `encrypted.txt`. You will devise a function named `messageFind` to perform this task. In the section named `Program design`, you will find the guideline to implement the function `messageFind`.

Complexity requirement: The input size is $O(n+m)$. So that your program should find out the message in $O(nm)$ worst-case time complexity. The space complexity must be $O(nm)$.

Task 2: Words separating

In this task, you will break the deciphered message of Task 1 into multiple words using the vocabulary, input from `dictionary.txt`. You will create another function named `wordBreak` to break the initial deciphered message into multiple words using the dictionary. To break the message into words, several cases can appear and discussed in the conversations which are as below:

- **Normal condition:** Break the message into available words
- **Use of large word:** If the combine of two or more words is also another large word in the dictionary and the message, use the large word. For an example, in the dictionary there are eight words: 'am', 'i', 'ice', 'cream', 'icecream', 'is', 'old', 'cold' and the message is 'icecreamiscold', Use 'icecream is cold' instead of other patterns.
- **Empty dictionary:** Whole message will be treated as a single word.
- **No matching words:** Whole message will be treated as a single word.
- **Partially matching words:** Dictionary may not contain all possible words. In this case, not matched segment of the message will be treated as separate words. It can appear in the beginning or in the middle or at the end of the message. For an example, in the dictionary there are five words: 'am', 'i', 'ice', 'cream', 'icecream' and the message is 'icecreamiscold', the task 2 will predict 'icecream i *scold*'. Let consider another example, in the dictionary there are six words: 'am', 'i', 'ice', 'cream', 'icecream', 'old' and the message is 'icecreamiscold', the task 2 will predict 'icecream i *sc* old'.

Keep in mind, words in the dictionary will be the combination of upper and lower case, so that case sensitivity is important here. You will find the guideline of the implementation of function `wordBreak`.

Complexity requirement: The worst case time complexity of function `wordBreak` should be $O(kM \cdot NM)$, where k is the size of input string i.e. message with no space. The space complexity is $O(kM)$.

Output

The output of your program of this text decipher problem will be as below:

```
The name of the file, contains two encrypted texts : encrypted.txt
The name of the dictionary file : dictionary.txt
---------------------------------------------------------------------
Deciphered message is Monashuniversity
True message is Monash university
---------------------------------------------------------------------
Program end
```

If the `dictionary.txt` is empty, the output should be exact same as below:

```
The name of the file, contains two encrypted texts : encrypted.txt
The name of the dictionary file : dictionary.txt
---------------------------------------------------------------------
Deciphered message is Monashuniversity
True message is Monashuniversity
---------------------------------------------------------------------
Program end
```

The words of `dictionary.txt` are not matching:

```
The name of the file, contains two encrypted texts : encrypted.txt
The name of the dictionary file : dictionary.txt
---------------------------------------------------------------------
Deciphered message is Monashuniversity
True message is Monashuniversity
---------------------------------------------------------------------
Program end
```

The words of `dictionary.txt` are partially matching (only 'Monash' is matched with the message):

```
The name of the file, contains two encrypted texts : encrypted.txt
The name of the dictionary file : dictionary.txt
---------------------------------------------------------------------
Deciphered message is Monashuniversity
True message is Monash university
---------------------------------------------------------------------
Program end
```

There is no `dictionary.txt` file exists:

```
The name of the file, contains two encrypted texts : encrypted.txt
The name of the dictionary file :
---------------------------------------------------------------------
Deciphered message is Monashuniversity
True message is Monashuniversity
---------------------------------------------------------------------
Program end
```

Note: Your program will be tested on the different set of inputs. The size of two texts may not be same. Even more, the name of the input files may be different.

Program design

You must design the solution of assignment using the concept of Object Oriented Programming. There will be a class named `Decipher` in `decipher.py` file. Functions of above mentioned tasks (`messageFind` and `wordBreak`) should be the functions of class `Decipher`. In class `Decipher`, you will use only one instance variable named `message`, which will be initialized with empty string.

Function: messageFind. As mentioned earlier, your `messageFind` function of class `Decipher` will find out the initial deciphered message (longest subsequence of common alphabets) from two encrypted text, input from the first input file (e.g. `encrypted.txt`). Therefore, the file name of the first input file will be only parameter of the function `messageFind`. This function will save the initial deciphered message into the class instance variable `message` and will not return anything. There is no user input or output required for this function.

Example of parameter, return, user input and user output of function `messageFind`
Parameter: **`'encrypted.txt'`**
Return: *No return*
User input: *No user input*
User output: *No user output*

Function: wordBreak. The function `wordBreak` of class `Decipher` will break the initial decrypted message (found by Task 1) into words using the dictionary, input from the second input file (e.g. `dictionary.txt`). So that the name of second file will be the parameter of this function. The function will not return anything, but save the words of the decrypted message in a single string form into the class instance variable `message`. There is no user input or output required for this function.

For an example: the contents of the dictionary is 'Monash', 'university', 'Melbourne', 'Ballarat', 'college', 'school' and the initial deciphered message, saved in the class instance variable `message` is 'Monashuniversity'. The function `wordBreak` will break the 'Monashuniversity' into two words: 'Monash' and 'university' and save these words in the form of a string in the class instance variable `message` as 'Monash university'. Keep in mind the words ordering should be maintained according to their appearance in the initial deciphered message.

Example of parameter, return, user input and user output of function `wordBreak`
Parameter: `dictionary.txt`
Return: *No return*
User input: *No user input*
User output: *No user output*

An accessor function. Finally, there will be an accessor function inside the class `Decipher`, named `getMessage`. This accessor function will return the current content of the class instance variable `message` as string. The function `getMessage` will not have any parameter, user input and user output.

Example of parameter, return, user input and user output of function `getMessage`
Parameter: *No return*
Return: `‘Monashuniversity’` (if the function is called immediately after task 1), **OR**
`‘Monash university’` (if it is called after task 2 and dictionary has all possible words), **OR**
`‘Monash university’` (if it is called after task 2 and dictionary has only word *Monash*), **OR**
`‘Monashuniversity’` (if the function is called after task 2 and either dictionary is empty or no existence of dictionary file), **OR**
empty string (if called after initialization)
User input: *No user input*
User output: *No user output*

User input and output: As mentioned earlier, none of the function of the class `Decipher` will take input from user or give output to user. Therefore, you will use `if __name__ == ‘‘__main__’’` module to take user input, to gives output and to use the class functions to get desired result. Keep in mind, this module is not the part of the class `Decipher`, but should be written inside `decipher.py` file.
Function decomposition: You can decompose any compulsory function into multiple small functions. It is optional. But make sure you are not changing the parameter and return of the compulsory functions: `messageFind`, `wordBreak` and `getMessage`.
Important: Any deviation from the above discussed design plan will effect the test scripts and your program will fail to pass.

Comments and docstring in the code

You should use following docstring format to prepare the docstring for your program:

```
def function name([parameters]):
     ‘‘‘
     Functionality of the function
     Time complexity:
     Space complexity:
     Error handle:
     Return:
```

```
    Parameter:
    Pre-requisite:
    '''
```

You can add line comments where required.

Crash prevention

It is the part of the assignment to make your program to be saved from any kind of crash/ interruption. It is good to use all possible kind of exception handlers to save your program from being crashed/ interrupted. If your program is crashed/ interrupted, the examiner/ assignment marker is not obliged to resolve the issue and/ or to mark the assignment. In this case, you will **zero**.

Use of in-built functions and data structures

If you decide to use in-built Python functions and structures, you must carefully consider their worst-case complexities. For example, inserting/retrieving an element in Python dictionary (which uses hashing) takes $O(n)$ in worst-case. This is because, as we will later see in the lecture, although hashing is quite efficient in practice, its worst-case complexity for insertion/retrieval is still $O(n)$. So DO NOT use hashing and python dictionary. It may not always be easy to determine the worst-case complexities of all in-built functions and structures. Therefore, it is strongly recommended that you use only the basic data structures (such as Python lists). This assignment can be easily completed using the basic data structures without using any advanced in-built functions.

`END`