FIT2004 S1/2019: Assessment questions for week 4

THIS PRAC IS ASSESSED! (7.5 Marks)

DEADLINE: Sunday, 24th March 2019 23:55:00 ADST
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**. This practical work will be marked on the time, space complexity and output of your program. You must add comments and documentation in your code by the guideline of commenting and documentation that you learnt in FIT1008/2085. Your program will pass through the python-developed test scripts. You have to make sure that your assignment works fine with different kinds of input. Demonstrators are not obliged to mark programs that do not run or that crash. In this case, you WILL get **0**.
SUBMISSION REQUIREMENT: You will submit a zipped file containing your Python program file (named `TextCount.py`) as well as a PDF file (named `Analysis.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 okay. 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 count helper

Recently, Melbourne Literature Society has announced that they are hunting for young story writer. The society has asked all young teenager to submit a story. 17 years old Jonathan is interested to participate, but he is a bit tensed. In this competition, there are some conditions. He needs a program to check his writing is meeting the conditions or not. From his school friend, he learnt that in this semester you are focusing on developing efficient algorithms and data structures. He really hopes that you can help him and has come to seek your help. (Please read the conversation very carefully and retrieve the tasks.)
You: “Hi Jonathan, How’s going?”
Jonathan:“Hi, I am ok.”. You said “Tensed?”
Jonathan:“Hmm”. “No worries, tell me how can I help you”- you said to Jonathan.
“You know MLS are hunting for young story writer and I have a dream to be a writer like JK Rowling. I love Harry Porter stories”.“I am going to send them my story, but they have some conditions”.“My story should meet these conditions, but I do not know how I am going to check these conditions in my writings”- Jonathan said.

You: “Ok, tell me the conditions, I will develop a fast and memory efficient program for you which will check your writing.”
Jonathan: “Thanks”. “The essay should be within 10000 words. No words will appear more than 50 times”. “So I need a program to find out the number of words in my story including each word count”.
You: “Well, that can be done”.
Jonathan : “But the word count shall not include the following auxiliary verbs: `am`, `is`, `are`, `was`, `were`, `has`, `have`, `had`, `been`, `will`, `shall`, `may`, `can`, `would`, `should`, `might` and `could`; and articles: `a`, `an` and `the`”. “It’s really easy task, but what types of punctuation marks are you going to use in our writing”- you asked.
“I will use only `comma(,)`, `period(.)`, `question(?)`, `exclamation(!)`, `colon(:)`, `semicolon(;)`, and `quotation(")`”
“Ok, I will check these punctuation marks in the program”.
“But there is more”-Jonathan. “Ok go ahead”- you.
“The program will show the list of the words, sorted in alphabetical order with their counts. Can I also ask you to find out the k top most frequent words appears that in my writing.”-Jonathan.
You: “Ok, consider it done”
“Thanks. You are great!” replied by Jonathan.
“By the way, when do you need this program?”-You. “I think I need this before 24 March 2019. I am sorry, it is very short time”-Jonathan.
You: “It’s ok”. “Thanks”-Jonathan. “No worries, bye”-You. Jonathan- “Bye”.
He is full of excitement and you do not want to disappoint him. You do not have much time, so start working on it now.

Input

The input file named `Writing.txt` consists of the variable number of words not sorted in alphabetical order. Each word consists of only lowercase English characters, i.e., there is no uppercase letter. Moreover, Sentences of the input are separated by *newline*.

```
an aggressive alien creature visited our planet.
it was ugly, with a big nose, pinkish hairy skin, and feet that smelled.
it was frightened of us for no reason.
it resented our differences and laid claim to our planet.
this strange alien was an earth human.
it called me ‘‘alien’’.
```

Input size. Let n be the total number of words in the writing and m be the maximum number of characters in a word. The upper bound of the input size is $O(nm)$.
Your program must be able to do several tasks as described below.

Task 1: Preprocessing

In this task, your program will preprocess the input from `Writing.txt` by removing the auxiliary verbs, articles and punctuation marks mentioned in the conversation between you and Jonathan. You will write a function named `preprocess` for this task. This function will take the *filename* of the input file as a parameter and will return *a list of words* after preprocessing.

Keep in mind: words are separated by whitespace characters (*space* (‘ ’), *tab* (‘\t’), and *newline* (‘\n’)).
Example of parameter and return values:
Parameter: `‘Writing.txt’`
Return: `[‘aggressive’,‘creature’,‘alien’,‘big’,....]`
Important: Any difference from the above mentioned function name, parameter and return type will effect the test scripts and your program will NOT pass.

Complexity requirement: The input size is $O(nm)$. So that your program should read and preprocess the input in $O(nm)$ worst-case time complexity. The space complexity must be $O(nm)$.

Task 2: Sorting the words

In this task, you will sort the preprocessed words in alphabetical order in faster way. You will create another function named `wordSort` to sort the all words in alphabetical order. The parameter of the function will be the *list of words* which are preprocessed in Task 1. The function will return the *sorted list of word* which are remained after preprocessing.
Example of parameter and return values:
Parameter: `[‘aggressive’,‘creature’,‘alien’,‘big’,....]`
Return: `[‘aggressive’,‘alien’,‘big’,‘creature’....]`
Important: Any difference from the above mentioned function name, parameter and return type will effect the test scripts and your program will fail to pass.

Complexity requirement: Let consider the size of preprocessed words is $O(nm)$. You need to develop the faster `wordSort` to sort the words. The worst case time complexity of function `wordSort` should be $O(nm)$. Keep in your mind, the space complexity must be $O(nm)$.

Task 3: Showing the number of total words including the frequency of each word

In task 3, you need to calculate and display the number of total words in the writing after preprocessing and the count of each word. To do this, you will write a function named `wordCount` to calculate and print the total number of words including the frequency (count) of each word. The parameter of the `wordCount` is the *sorted list of words* of Task 2. This function will print the total number of words and each word’s counts. Finally, it will return a list with two values: (a) the total number of words and (b) a list of words with their count. The list of words should be remain sorted.
Example of parameter and return values:
Parameter: `[‘aggressive’,‘alien’,‘big’,‘creature’....]`
Return: `[44,[‘aggressive’,1],[‘alien’,3],[‘big’,1],[‘creature’,1],....]`
Important: Any difference from the above mentioned function name, parameter, return type and ordering of return values will effect the test scripts and your program will NOT pass.

Complexity requirement: Let consider, the number of preprocessed words is n. To count the number of each word, the worst case time complexity of your function should not be more than $O(nm)$. The space complexity should also be $O(nm)$.

Task 4: Showing the k top most words appears in the writing

This is your final task in this assignment. In this task, you have to find out and display the k top-most frequent words in the writing after removing auxiliary verb and articles. You will construct a method named `kTopWords` to find out and display the k top most frequent words in the writing. There are two parameters of the function: (a) The value of k and (b) the list of sorted words with their frequencies which will be similar like: `[['aggressive',1],['alien',3],['big',2],....]`, where the first element of the list is the word and second element is it's count. The function will return similar list of its parameter, but will only contain the k number of top-most frequent words. Keep in your mind: if the count of two words are same, the word comes earlier in the sorted list will have higher priority over another.

Example of parameter and return values:

Parameters: 4 and `[['aggressive',1],['alien',3],['big',1],['creature',1],....]`

Return: `[['it',4],['alien',3],['our',3],['and',2]]`

Important: Any difference from the above mentioned function name, parameters, ordering of parameters and return type will effect the test scripts and your program will NOT pass.

Complexity requirement: To find the k top most words, the worst case time complexity of your function should not be more than $O(n \log k)$. The space complexity should also be $O(km)$.

Output

The output of your program of the above shown input will be as below:

```
Words are preprocessed..
Do I need to display the remaining words: Y
aggressive
alien
creature
visited
our
planet
it
ugly
with
big
nose
pinkish
hairy
skin
and
feet
that
smelled
it
frightened
of
us
for
```

```
no
reason
it
resented
our
differences
and
laid
claim
to
our
planet
this
strange
alien
earth
human
it
called
me
alien

The remaining words are sorted in alphabetical order
Do you want to see: Y
aggressive
alien
alien
alien
and
and
big
called
claim
creature
differences
earth
feet
for
frightened
hairy
human
it
it
it
it
laid
me
no
```

```
nose
of
our
our
our
pinkish
planet
planet
reason
resented
skin
smelled
strange
that
this
to
ugly
us
visited
with

The total number of words in the writing: 44
The frequencies of each word:
aggressive : 1
alien : 3
and : 2
big : 1
called : 1
claim : 1
creature : 1
differences : 1
earth : 1
feet : 1
for : 1
frightened : 1
hairy : 1
human : 1
it : 4
laid : 1
me : 1
no : 1
nose : 1
of : 1
our : 3
pinkish : 1
planet : 2
reason : 1
resented : 1
```

```
skin : 1
smelled : 1
strange : 1
that : 1
this : 1
to : 1
ugly : 1
us : 1
visited : 1
with : 1

How many top-most frequent words do I display: 4
4 top most words appear in the writing are:
it : 4
alien : 3
our : 3
and : 2
```

If the `Writing.txt` is empty or there is no word remaining after preprocessing, the output should be exact same as below:

```
Unable to continue:
1. Writing.txt is empty or
2. There is no word remaining after preprocessing.
```

Note: Your program will be tested on a different writing file, i.e., different types of writing with different number of words.

Things to note

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.

```
-=o0o=-
  END
-=o0o=-
```