Assignment 4

420-ENM-MT Algorithm, Pseudo-code and Design

Due: February 21st 2019 before class

Instructions

- This assignment is worth **5% of your final grade**;
- You must submit three (3) files on **Omnivox**. Submit theory questions as a single **pdf** and coding questions as `.py` files;
- **Read every question carefully**;
- Some problems require you to write code. Make sure to read the instructions and submit the corresponding `.py` files with your assignment;
- Your code will be tested with **Python 3.7**;
- Using online resources is allowed, but you should cite your sources for any algorithm or code you did not write yourself. You must prove you understood the solution by properly commenting it, otherwise you will be given a grade of zero for **plagiarism**.

Problem 1 (10 points)

In class, we defined a hash function which associates each letter to its position in the alphabet and sums the letters of a word.

a → 1
b → 2
…
y → 25
z → 26

Using this hash function, **draw what the hash table looks like** in the example below **after each operation**.

```
table = empty hash table of size 6
table[“Olivier”] = 450-111-1111
table[“Sakshi”] = 514-222-2222
table[“Johny”] = 438-333-3333
```

Problem 2 (10 points)

In a previous assignment, you were asked to write an algorithm to find duplicates across two lists of numbers.

Since this algorithm relied on sorting, it could not be better than `O(n log n)`.

It is time to see if hash tables can do better.

Write a **Python** function `find_duplicates(items1, items2)` which takes two lists of values as argument and returns a list containing all values which were present in both lists.

The function must work both for lists of strings and list of integers.

Examples

- `find_duplicates([1, 2, 3], [2, 4, 5])` → `[2]`
- `find_duplicates(["dog", "cat"], ["cat", "giraffe"])` → `["cat"]`

The algorithm must be `O(n)`, but you are allowed to use any Python data-structure or standard library. In particular, you are encouraged to use Python hash table type: `dict`.

Answer this question by filling in the file `find_duplicates.py`, then submit it with your assignment.

Problem 3 (20 points)

Nowaday, we have access to enormous quantities of data: news articles, online books, blogs, etc. Due to the amount of data available, computers are a needed to analyse text and its content.

One basic way to analyse a text is to find the number of occurence of each word. Even though this does not provide the full meaning, this is a useful tool to quickly identify the themes and origin of the text.

Write a **Python** function `count_words(text)` which takes a string as argument and returns the count of each word found in the string.

The lists can contain anything, including integers and strings.

Example:

```
txt = 'Text analytics derives information from the text.'

count_words(txt)
```

Above should output the following `dict`:

```
{'analytics': 1,
 'derives': 1,
 'from': 1,
 'information': 1,
 'text': 2,
 'the': 1}
```

You are allowed to use any Python data-structure or standard library. In particular, you are encouraged to use Python hash table data-structure: `dict`.

Notice that before counting words, your input will need to pass through a process called **standardizing**. This means removing punctuation and capital

letters and splitting your text into words. Methods such as `text.lower()` or `text.split()` may come useful.

Answer this question by filling in the file `count_words.py`, then submit it with your assignment.