Middle East Technical University

Department of Computer Engineering

CENG 111

Fall 2016

Take Home Exam 3

Version 2.0

REGULATIONS

Due date: 23:59, 03 January 2017, Tuesday *(Not subject to postpone)*

Submission: Electronically. You should save your program source code as a text file named `the3.py` and submit it to us via the course's COW page.

Team: There is **no** teaming up. This is an EXAM.

Cheating: Source(s) and Receiver(s) will receive zero and be subject to disciplinary action.

INTRODUCTION

In THE3 you will be doing an introductory level Random Natural Language Generation. Our domain language will be Turkish because of its suitability for the technique we are going to use.

The Turkish language has a strong agglutinative nature with a highly algorithmic syllable formation. In support of this the hyphenation of a word into its syllable is also quite algorithmic.

The program you are going to write will work through a relatively large corpus (a fairly large collection of Turkish sentences). The task is to break down each word into its syllables. Then the statistics (counts) of which syllable follows which syllable is collected over all the words. Here the inter-word space and the sentence ending period can also be considered as a syllable.

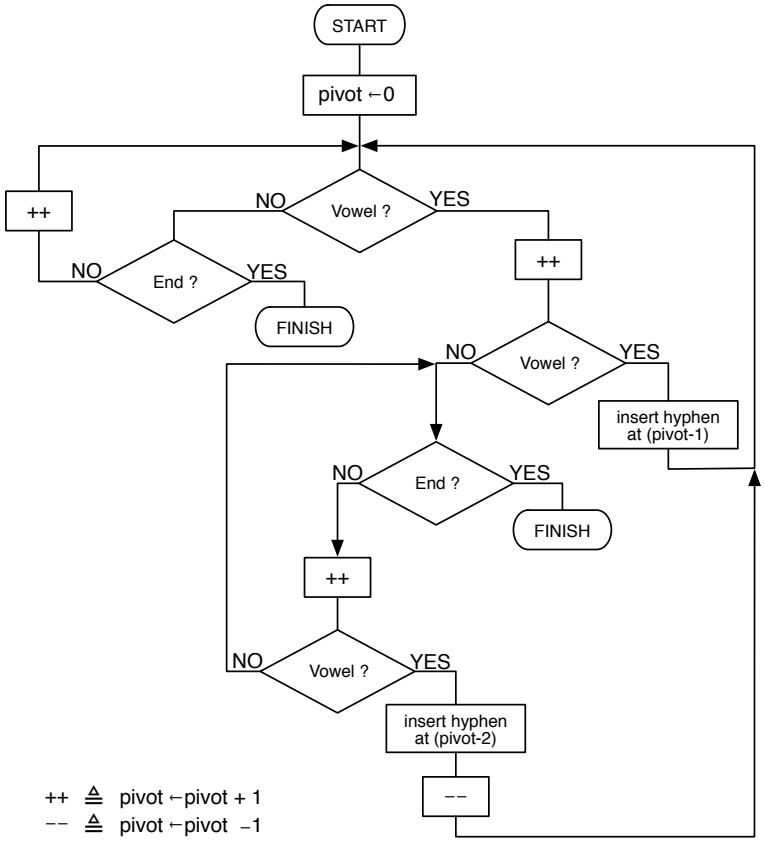
The task of generating an artificial "Turkish imitating" text will start with a randomly chosen syllable. Then one of the two highest scoring probable successor syllables will be appended to this syllable (it is also possible that one of these two syllables is an inter-word space –so the created word will terminate). Which one of the two highest is to be used is chosen randomly.

Basically the process will stop after n words (n is an input parameter).

(There is another termination criterion introduced below).

PROBLEM & SPECIFICATIONS

Study the below described hyphenation algorithm. Understand it. Then encode it in Python. You are going to name the function that takes as argument a word to be hyphenated as `hyphenate`. Do not choose another naming. `hyphenate` will return a list of syllables.



Here is an example:

```
>>> hyphenate("saatCigillerdenmiS")
["sa","at","Ci","gil","ler","den","miS"]
```

As you might have realised, capital letters stand for the Turkish special characters. So, `U` is representing the Turkish letter ü, `I` is the Turkish letter ı, `G` is the Turkish letter ğ and so on.

The following sequence of tasks (starting with the next sentence, below) including outputting of your generated text, shall be carried out when a function `execute()` is called. This function `execute()`, that you will be defining, will have no arguments.

The next subtask is to read the input line by line. Break down each line into its words (there are nice string functions in Python to do that).

You will make a counting record of successive syllables. So, for example, if `'ke'` is followed by `'di'` 3 times, and by `'dir'` 4 times you will be knowing this for a later retrevial.

Of course it is wise to update a data structure with each processing of a word in a line. We propose to use a 'dictionary of dictionaries' data structure in Python.

After all the data is processed, it is time to determine, for each syllable, the first and second most frequently occurring follower (some two syllables). Note that since 'space' and 'period' is also treated as a syllable, for some syllables they can be among the winners, as well (can you think of which syllables these might be?)

Now you are supposed to pick randomly any syllable which can follow a period or a space (attn: for the first and last time you take any syllable and not one of the most frequently occurring twos). By this syllable you start the formation of your output text. By random choices that you will perform among the possible two followers you will pick and suffix it to the last one. You will carry on in this manner until

- either you have reached n number of words (this expected count is given in the input)
- or your output text exceeded m characters.

in both cases you shall complete the last syllable affixation.

As you have guessed already, possible spaces will terminate a word, possible periods will terminate a sentence.

- The input is from the standard input and the output is to the standard output.
- The first input line contains two integers n and m. n is the expected count of words to be formed in the output. m is the maximal count of characters of the output.
- All the following lines before the last line are the input text.
- The last line consists of a single character: An equal sign:
  `=`
  (This is to help you with the termination criteria for the text read looping)
- Producing more words than n and/or characters more than m will not cause a grade loss.
- The input text is lowercase only (the capitalisation rules of Turkish writing does not apply). Upper case letters always represent Turkish special characters.
- Input text does not contain word breaks over line endings. Also your output shall not.
- The only punctuation that will appear is the sentence-ending dot.
- Words are separated exactly by one blank.
- Line starts and endings do not contain blanks.
- The last line ends with a period.
- A word is any sequence of letters terminated by a space or period.
- Neither in the input nor the output a period will not be followed by a space.
- The first syllable of the input text shall be assumed to be following a blank (so it shall contribute to the statistics of the syllables that follow a blank). This blank will not be present in the input.
- There is no limit of the count of syllables that form a word.
- There is no limit for the count of characters in a line, neither in the input nor the output. You are allowed to break your output into lines.
- Both the names for the functions `hyphenate` and `execute` shall not be changed.
- You are allowed to define as many helper functions as you desire.
- Do not code any input/output action outside a function definition.
- In your submitted code do not call `execute()`. This is the job of the evaluation script.
- You are expected to follow the above given algorithm. To not even think of using snippets you may find on the web. That will be considered cheeting in the exam.

GRADING

Comply with the specifications. Do not try to beautify neither the input nor the output. Your program will be graded through an automated process.

Your program will be tested with multiple input texts (a distinct run for each `execute` call. But multiple `hyphenate` calls can be performed in a single run).

Any program that performs only 30% and below will enter a glass-box test (eye inspection by the grader TA). The TA will judge an overall THE3 grade in the range of [0,30]. The glass-box test grade is not open to discussion nor explanation. 50% of the grade will consist of the testing of the function `hyphenate`. The remaining 50% is due to the testing of the function `execute`.