CS 5134/6034 Natural Language Processing
Assignment #1
Due: Monday, Sept 16, 2024 by 9am (before class)

Your task is to build a morphological analyzer from scratch. Your morphological analysis program should accept three input files:

1. a dictionary file
2. a rules file
3. a test file

Your program should be named "morphology.py" and should accept 3 files as command-line arguments in the order above. Running your program in command-line must strictly follow:

```
python3 morphology.py [dictionary file] [rules file] [test file]
```

For example, running your code on our sample files should look like this (do NOT hard-code the file names; they will be different in grading):

```
python3 morphology.py Dict0.txt Rules0.txt Test0.txt
```

You should only submit one source code file "morphology.py" to Gradescope.

The Dictionary File

The dictionary file will consist of words paired with their possible parts-of-speech (POS). If a word can have multiple parts-of-speech, then each POS will be on a separate line. The POS tags should be treated as arbitrary strings (i.e., you should *not* assume a finite set of tags), but you can assume that all POS tags will be a single word (i.e., not a phrase). Some dictionary entries for irregular word forms may also be followed by a "ROOT" keyword and the root word. For example, the word "sat" is an irregular verb so a dictionary entry for "sat" would include the "ROOT" keyword followed by the root word "sit".

Here is a sample dictionary file:

```
carry  verb
furry  adjective
marry  verb
bark   noun
bark   verb
sit    verb
sat    verb ROOT sit
```

Your program should treat the dictionary as **case insensitive** for matching (e.g., the words "dog", "Dog", and "DOG" should all be considered the same word). But you should keep the original

form for output. You can assume that any root listed in the dictionary will have its own entry in the dictionary (e.g., "sit" is the root of "sat" in the example above and has its own dictionary entry). You can also assume one word and its POS tag will uniquely determine a ROOT in the dictionary. You should **not** assume that the dictionary file is sorted in any way.

The Rules File

The rules file will consist of morphology rules, one per line. Each rule will have 6 parts:

1. rule ID
2. PREFIX or SUFFIX keyword
3. beginning or ending characters (relative to the prefix/suffix keyword)
4. replacement characters, or a hyphen if no characters should be replaced
5. the part-of-speech tag **required for the originating word** from which the new word is derived
6. the part-of-speech tag that **will be assigned to the new (derived) word**

The characters `->` will be used to separate the POS tag required for the originating word from the POS tag that will be assigned to the derived word. Each rule will end with a period. A sample rule file is shown below:

```
1 SUFFIX  ly   -  adjective   ->   adverb .
2 SUFFIX  ily  y  adjective   ->   adverb .
3 SUFFIX  ed   -  verb   ->   adjective .
4 SUFFIX  ed   -  verb   ->   verb .
5 SUFFIX  ied  y  verb   ->   adjective .
6 SUFFIX  ied  y  verb   ->   verb .
7 PREFIX  re   -  verb   ->   verb .
8 SUFFIX  s    -  noun   ->   noun .
9 SUFFIX  s    -  verb   ->   verb .
```

The Test File

The test file will contain words that your morphological analyzer should be applied to. The file will contain one word per line. A sample test file is shown below:

```
carry
carried
remarried
xyz
```

Morphology Rule Matching Algorithm

Given a test word, first look up the word in the dictionary. If the word is present, then its dictionary definition should be returned and the process ends. *No morphological analysis should be done in this case.*

If the test word is not in the dictionary, then apply the morphology rules. When applying the rules, remember that (1) ALL of the rules must be applied because more than one rule may be successful, and (2) the rules must be applied recursively until a match is found in the dictionary OR until no more rules can be applied. When you find a word in the dictionary that matches a rule, that particular derivation path stops and is deemed successful. But remember that you still need to search exhaustively for additional derivations.

If your morphological analyzer cannot find any derivation for a word, then your program should generate a default definition for the word as a "noun". (In practice, unknown words are typically assumed to be nouns because unfamiliar words are often proper names.)

Multiple Derivations: Multiple derivations for a word will be common! For example, if a word ends in "ied" then the 3rd, 4th, 5th, and 6th rules in the sample rules (shown on the previous page) may all apply! *Every distinct definition* that is produced should be printed. For example, if "carried" can be derived as both a verb and an adjective, then both the verb and adjective definitions should be printed. However, if your program generates exactly the same definition (as defined in the following section including WORD, POS, PATH, etc.), then only print that definition once (i.e., there should not be duplicate rows in your output).

Output Specifications

Your program should print to the screen. Each *word definition* should be in the following format:

`WORD=<word> POS=<pos> ROOT=<root> SOURCE=<source> PATH=<path array>`

There is exactly **one tab** (not spacebar) between each item. Each definition should appear on a separate line.

ROOT should be the word ultimately found in the dictionary during the application of morphology rules. If the word in the dictionary has an explicit root defined, then use its defined root (e.g., use "sit" as the root for "sat"). If no explicit root is defined, then assume that the word itself is a root form.

SOURCE indicates how the definition was obtained, and has 3 possible values:

1. **dictionary**: if the word was found in the dictionary *without* applying morphology rules.
2. **morphology**: if the morphological analyzer successfully derived the word.
3. **default**: if the word is not in the dictionary and the morphological analyzer failed to find a derivation for the word.

PATH is a comma-separated list of rule ids, indicating the rules that were used. The PATH should begin with the rule that was applied to the word found in the dictionary and then list the other

rules in the order that they were subsequently applied. If the SOURCE is dictionary or default, then print "PATH=-". If multiple derivations are found using the same set of rules but in a different order, these are considered to be **different** derivations with distinct paths.

Remember that the morphological analyzer can generate multiple definitions for a word. You should print each derivation on a separate line. Please print all possible definitions of one given word **in the alphabetic order** based on their PATH values. If multiple derivations have the same PATH, the sort those derivations alphabetically based on the POS tag. Please print a blank line between the sets of derivations for different words.

For example, the words "carry", "carried", "remarried" and "xyz" should produce the following output (based on the sample dictionary and rule files shown earlier):

```
WORD=carry POS=verb ROOT=carry SOURCE=dictionary PATH=-

WORD=carried POS=adjective ROOT=carry SOURCE=morphology PATH=5
WORD=carried POS=verb ROOT=carry SOURCE=morphology PATH=6

WORD=remarried POS=verb ROOT=marry SOURCE=morphology PATH=6,7
WORD=remarried POS=adjective ROOT=marry SOURCE=morphology PATH=7,5
WORD=remarried POS=verb ROOT=marry SOURCE=morphology PATH=7,6

WORD=xyz POS=noun ROOT=xyz SOURCE=default PATH=-
```

Grading Criteria

1. Please use Gradescope to submit **one** source code named "morphology.py". Your program will be graded based on **new test cases**! So please test your program thoroughly to evaluate the generality and correctness of your code! Even if your program works perfectly on the examples that we give you, that does not guarantee that it will work perfectly on new test cases.

2. Please exactly follow the formatting instructions specified in this assignment. **You will get a 0 grade if you fail to follow the specifications.**

When you submit your code on Gradescope, you can get an instant score from Autograder based on the examples given to you so you can make sure your output has the correct format and "get a sense of" what grades your program may receive. After the due date, we will use new test cases to re-evaluate your program. Besides Gradescope, we also provide a grading script "grading.sh" that will help you compare your output with the true answer. The "grading.sh" should be put in the same directory as your "morphology.py" and other example input files. Then run "bash grading.sh". If you are using Linux or Mac OS, it should all work smoothly. If you are using Windows, you need a bash shell first. For example, install Windows Subsystem for Linux (WSL)[1] and run the same commands.

3. You can **NOT** use any external software packages or dictionaries to complete this assignment except for **Python3.9 Standard Library**.[2] For example, libraries for general I/O handling, math

[1] `https://learn.microsoft.com/en-us/windows/wsl/install`

[2] `https://docs.python.org/3.9/library/index.html`

functions, and regular expression matching are ok to use. The Gradescope Autograder will raise an error and give you zero when you import external packages.

4. You can **NOT** use any generative AI tools such as ChatGPT for this homework. All submitted code **must be your own**.