CS-5340/6340, Programming Assignment #1
Due: Wednesday, September 15, 2021 by 11:59pm

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, and (3) a test file. Your program should be named **"morphology"** and should accept these files as command-line arguments in the order above. For example, if you use Python then we should be able to run your program like this:

python3 morphology.py <dictionary_file> <rules_file> <test_file>

Running your code on our sample files should look like this:

python3 morphology.py sampleDict.txt sampleRules.txt sampleTest.txt

If you use Java, you should invoke Java similarly and be sure to accept the same arguments on the command-line in the same order.

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 that a word and its POS tag will uniquely determine its ROOT (i.e., the dictionary will not contain multiple instances of the same word and POS tag but with different roots). 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 identifier (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 rules 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
```

Morphological Analysis

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 using the method described in class (and on the lecture slides). All legal derivations for the test word should be returned.

If your morphological analyzer cannot find any derivation for a word, then your program should generate the part-of-speech tag "noun" as a default value. [In practice, unknown words are typically assumed to be nouns because many unfamiliar words are 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 derivation* 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.

OUTPUT SPECIFICATIONS

Your program should output each *word derivation* in the following format:

`WORD=<word> POS=<pos> ROOT=<root> SOURCE=<source> PATH=<rule list>`

There should be at least one space between each item. Each derivation should appear on a separate line.

ROOT should be the word ultimately found in the dictionary during the application of the rules. If the dictionary word 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:

> **dictionary**: if the word was found in the dictionary *without* applying morphology rules.
> **morphology**: if the morphological analyzer successfully derived the word.
> **default**: if the word is not in the dictionary and the morphological analyzer failed to find any 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=-".

Important: If multiple derivations are found using the same set of rules but in a different order, these are considered to be <u>different</u> derivations with distinct paths! You should print each derivation on a separate line. Please print the derivations in alphabetic order based on their PATH values. If multiple derivations have the same PATH, then 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

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.

Please exactly follow the formatting instructions specified in this assignment. We will deduct points if you fail to follow the specifications because it makes our job much more difficult to grade programs that do not conform to the same standards.

IMPORTANT: You may not use ANY external software packages or dictionaries to complete this assignment except for *basic* libraries. For example, libraries for general I/O handling, math functions, and regular expression matching are ok to use. But you may not use libraries or code from any NLP-related software packages, or external code that performs any functions specifically related to morphological analysis. All submitted code *must be your own.*

SUBMISSION INSTRUCTIONS

Please use CANVAS to submit a gzipped tar file named "morphology.tar.gz". (This is an archived file in "tar" format and then compressed with gzip. Instructions are given on the next page if you're not familiar with tar files.) Your submission should contain the following items:

1. The source code files for your program. Be sure to include all files that we will need to compile and run your program!

   REMINDER: your program **must** be written in Python or Java, and it **must** compile and run on the Linux-based CADE (lab1 or lab2) machines! We will not grade programs that cannot be run on the Linux-based CADE machines.

2. An executable *shell script* named `morphology.sh` that contains the exact commands needed to compile and run your morphology program on the data files provided on CANVAS. We should be able to execute this file on the command line, and it will compile and run your code. For example, if your code is in Python and does not need to be compiled, then your script file might look like this:

   ```
   python3 morphology.py sampleDict.txt sampleRules.txt sampleTest.txt
   ```

   If your code is in Java, then your script file might look something like this:

   ```
   javac Morphology/*.java
   java Morphology/MainClass sampleDict.txt sampleRules.txt sampleTest.txt
   ```

3. A `README.txt` file that includes the following information:

   - Which CADE machine you tested your program on
     (this info may be useful to us if we have trouble running your program)
   - Any known idiosyncracies, problems, or limitations of your program.

4. Submit one trace file called **morphology.trace** that shows the output of your program when given the sample files provided on Canvas.

HELPFUL HINTS

TAR FILES: First, put all of the files that you want to submit in a directory called "morphology". Then from the parent directory where the "morphology" folder resides, issue the following command:

tar cvfz morphology.tar.gz morphology/

This will put everything underneath the "morphology/" directory into a single file called "morphology.tar.gz". This file will be "archived" to preserve any structure (e.g., subdirectories) inside the "morphology/" directory and then compressed with "gzip".

FYI, to unpack the gzipped tar file, move the morphology.tar.gz to a new location and issue the command:

tar xvfz morphology.tar.gz

This will create a new directory called "morphology" and restore the original structure and contents.

For general information on the "tar" command, this web site may be useful:
https://www.howtogeek.com/248780/how-to-compress-and-extract-files-using-the-tar-command-on-linux/

TRACE FILES: You can generate a trace file in (at least) 2 different ways: (1) print your program's output to standard output and then pipe it to a file (e.g., `python3 morphology.py sampleDict.txt sampleRules.txt sampleTest.txt > morphology.trace`), or (2) print your output to standard output and use the unix *script* command before running your program on the test files. The sequence of commands to use is:

```
script morphology.trace
python3 morphology.py sampleDict.txt sampleRules.txt sampleTest.txt
exit
```

This will save everything that is printed to standard output during the session to a file called `morphology.trace`.