Advanced - Regex

EECS 201 Winter 2022

Submission Instructions

This assignment will be submitted as a repository on the UMich GitLab server. Create a private, blank, **README-less (uncheck that box!)** Project on it with the name/path/URL `eecs201-adv-regex` and add `brng` as a Reporter. The submission branch will be `latin`. If this branch is not already the default initial branch, you initialize the local repo with an additional argument: `git init --initial-branch=latin` if your version of Git is recent enough. Otherwise you can create a branch with this name after your first commit. The repository should have the following directory structure, starting from the repository's root:

```
/
|-- report.txt
|-- pig-latinfy.sh
```

Preface

I highly suggest that you do this homework in a Linux environment, be it WSL on Windows (on the Linux filesystem, not the Windows filesystem), an Ubuntu virtual machine, or the class server. The reasoning for this is that some tools that deal with regular expressions (namely for this assignment: `sed` and `grep`) may differ in behavior depending on *nix system. Linux systems use GNU `sed` and `grep` while mac OS (and FreeBSD) use BSD `sed` and `grep` which have some subtle differences in behavior. In particular, GNU `sed` on Linux has some helpful extensions that can deal with a particular case.

In this assignment you'll be provided yet another zipped archive containing some starter empty files and scripts. Use your preferred tool to retrieve this file and extract it (see Basic 2 if you need a review).

`https://www.eecs.umich.edu/courses/eecs201/wn2022/files/assignments/adv-regex.tar.gz`

Initialize a Git repository in the extracted `adv-regex` directory, following the submission instructions.

1 More `sed` fun (10)

Pig Latin is a language game in which English words are made to sound like a faux-Latin by moving around groupings of letters.
Here are the basic rules for this exercise's variant:

1. For words that begin with a single consonant before running into a vowel, the consonant is moved to the end and "ay" is added after.
   - "<u>h</u>ello" = "ello<u>h</u>ay"
   - "<u>c</u>at" = "at<u>c</u>ay"
   - "<u>l</u>iquid" = "iquid<u>l</u>ay"
2. For words that begin with a multiple consonants before running into a vowel, the consonant group is moved to the end and "ay" is added after.
   - "<u>str</u>ing" = "ing<u>str</u>ay"
   - "<u>fr</u>iend" = "iend<u>fr</u>ay"
   - "<u>wr</u>ong" = "ong<u>wr</u>ay"
3. For words that begin with a vowel, "yay" is simply added at the end.
   - "I" = "I<u>yay</u>"

   - "apply" = "apply<u>yay</u>"
   - "income" = "income<u>yay</u>"
4. For this Pig Latin variant, the letter 'y' is always a consonant. For your convenience, consonants are BCDFGHJKLMNPQRSTVWXYZ and vowels are AEIOU.
5. If the word contains no vowels, then leave the word alone.

For this exercise, write an **executable shell/Bash** script called `pig-latinfy.sh` that takes **its input on standard input** and turns it into Pig Latin and outputs the translation **on standard output**. **Be sure to set the shebang appropriately!** If you are on WSL and doing this on the Windows filesystem, be sure to have Git set the execute bits (look up how to do this)! The big caveats are:

- Each word on a line will be translated.
- Words with a punctuation mark adjacent will be translated and retain the punctuation mark. For example: "goodbye!" = "oodbyegay!"
- Capitalization of the word is preserved. For example: "My name is John" = "My amenay isyay Ohnjay".
- Hyphenated words consider each component to be a separate word. For example: "part-time" = "artpay-imetay".
- Spacing will be preserved. For example: "hello   world" = "ellohay   orldway"

The `adv-regex` directory has example text and their translated versions for reference.

Helpful hints

- There are other positional matches beside '`^`' and '`$`': the GNU `grep` manual documents some.
- If you find yourself making loops and if-conditions, you might be overcomplicating things.
- `sed` is already capable of reading from standard input without your intervention: you don't need to do anything special to read input. Simply running `sed` without an input file will cause it to read from standard input.
- GNU `sed` has extensions that deal with upper/lower case conversion. Its manual's section on the `s` command is enlightening.
- `diff` is able to show differences in files. Combine it with a process substitution (see the Shell Lecture) and you have an easy way to test your script...

2 Conclusion

1. Add and commit any changes you intend to submit.
2. Create a file called `report.txt`.
3. On the first line provide an integer time in minutes of how long it took for you to complete this assignment.
4. On the second line and beyond, write down what you learned while doing this assignment. If you already knew how to do all of this, put down "N/A".
5. Add and commit this `report.txt` file.