

Assignment 2 Instructions

CISC 220, Fall 2018

Administrative Notes: All of the administrative notes from Assignment 1 apply to this assignment as well. I will just summarize here and you can refer back to the Assignment 1 instructions for more details:


- No group work allowed
- Hand in your files to OnQ in the “Assignment 2” Dropbox area
- Follow instructions carefully or you will lose points
- Use the required names (including the “.sh” extension for shell scripts so that OnQ will accept them)
- There is a 24-hour “grace period” after the official deadline for each assignment. Submissions made within this period lose 10% but are still accepted. After that, no submissions accepted (except for students with accommodations).

Deadline: The deadline for this assignment is **Friday, October 12 at 8 a.m.**

Part 1: A Grade Distribution Script

Suppose you are working as a prof or TA. Your students have an assignment which requires them each to hand in exactly one file. You've got a directory with all of the submitted files in it, and no other files. You're also lucky enough to have a script that will do all the work of marking a submission; it will take the name of a submission file as its only parameter and will print out a single upper-case letter (A, B, C, D or F) as its only output.

You'd like to know the distribution of marks in your class. Write a bash script called `gradeDist.sh` (the “.sh” extension is required!) that will take two parameters:

- 
1. The name of your marking script
 2. The name of the folder containing the assignment submission files

Your `gradeDist.sh` script must call the marking script to mark each file inside the folder and print a report showing the number of files earning each of the letter grades. **If there are sub-folders inside the folder, your script must ignore them (and all of the files inside them).** The only output of your `gradeDist.sh` script should be a list of how many submission files received each of the five possible marks.

Please look at the example transcript later in this document for an example of the output format.

Part 2: Pig Latin

Write a bash script called `pigLatin.sh` that will take one parameter. You may assume that the parameter is a string of lower-case letters. Your script must print the equivalent of that word in Pig Latin.

There may be some of you who are unfamiliar with Pig Latin, and the details of Pig Latin vary with the user, so here are the rules of Pig Latin for this assignment:

- If the word starts with a vowel (a,e,i,o or u), just add "way" to the end of the word. For example: apple in Pig Latin is "appleway" and egg is "eggway".
- If the word starts with one or more consonants (before the first vowel) move those consonants to the end of the word and add "ay". For example, "dog" becomes "ogday", "street" becomes "eetstray", "computer" becomes "omputercay" and "gradual" becomes "adualgray".
- The letter "y" in English sometimes functions as a consonant and sometimes as a vowel. For the purposes of this assignment, let's assume that "y" at the start of a word is always a consonant. (I can't actually think of a counter-example, but there may be one out there; don't worry about it!) So "yellow" becomes "ellowyay" and "you" becomes "ouyay"
- You may not assume any limit on the number of consonants that might appear at the beginning of the word, even though the limit in English is 3. So you should accept a word like `brzkleet` and translate it to `eetbrzklay`.
- You may assume that every word contains at least one vowel.

If you grew up with different rules for Pig Latin, use the above rules for this assignment anyway.

Error Handling

Your scripts must be able to **handle the following four kinds of errors**:

- the `gradeDist.sh` script is called with a number of arguments other than two
- the first argument to `gradeDist.sh` is not the name of an existing, executable file
- the second argument to `gradeDist.sh` is not the name of an existing directory
- the `pigLatin.sh` script is called with a number of arguments other than one

If any of these kinds of errors occur your scripts must print an informative error message to the standard error stream and exit with a non-zero exit status. (You can choose any exit status number you want as long as it's not zero!) By "informative", I mean it must say enough to **tell the user which one of the four kinds of errors has occurred -- for example "error: wrong number of arguments" or "error: script needs two arguments"** instead of just "error" or even something vague like "argument error".

If no errors occur your scripts must exit with an exit status of zero. See the sample run included later for an example of what that would look like.

Your scripts may ignore the possibility of any other kinds of errors -- for example, you don't have to worry about what happens if the marking script named by the first parameter prints something other than A, B, C, D or F.

A Sample Test Run Of My Solution:

Warning: This is provided as an illustration. When marking your assignments we will use different test cases, so it's important that you try a variety of cases for each script to make sure that it works under all circumstances.

To test the gradeDist.sh script I created a directory containing 100 files. The names and sizes of the files were generated randomly. I also created two trivial marking scripts. The first (called grade1) marks files based on their size. The second (called grade2) marks files based on their names. A copy of the directory and the two marking scripts are posted on </cas/course/cisc220/assn2Examples>, so if you want to you can copy or refer to them and see if your own gradeDist.sh script gets the same results as mine.

```
-----$ ls
FILES/ grade1* grade2* gradeDist.sh* pigLatin.sh* randomFile*
-----$ gradeDist.sh grade1 FILES
A: 9
B: 17
C: 45
D: 18
F: 11
-----$ gradeDist.sh grade2 FILES
A: 7
B: 20
C: 41
D: 17
F: 15
-----$ gradeDist.sh singleArg 2>errorMessage.txt
-----$ echo $?
1
-----$ cat errorMessage.txt
error: gradeDist.sh needs two arguments
-----$ gradeDist.sh noSuchScript FILES
error: noSuchScript is not an existing, executable file
-----$ echo $?
2
-----$ gradeDist.sh grade1 NO-SUCH-DIR
error: folder NO-SUCH-DIR does not exist
-----$ echo $?
2
-----$ pigLatin.sh dog
ogday
-----$ pigLatin.sh cat
atcay
-----$ pigLatin.sh trouble
oubletray
-----$ pigLatin.sh structure
ucturestray
-----$ pigLatin.sh egg
eggway
-----$ pigLatin.sh illustrate
```

```
illustrateway
-----$ pigLatin.sh brchlect
ectbrchlay
-----$ pigLatin.sh o
oway
-----$ pigLatin.sh m
may
-----$ pigLatin.sh one two three
error: wrong number of arguments for pigLatin.sh (needs exactly one)
```

What to hand in to the Assignment 2 Dropbox area: All you need to hand in are your two scripts: `gradeDist.sh`, and `pigLatin.sh`. Please use those exact names or we will give you an administrative penalty. We may be using marking scripts to help us evaluate your scripts and if we have to stop and re-name scripts for lots of students it really slows down the process.

Marking Scheme:

- `gradeDist.sh` (except for error handling): 5
- `pigLatin.sh` (except for error handling): 5
- error handling: 5
- total: 15

The error handling points are 1 point for each of the 4 kinds of error messages required and 1 additional point for sending the messages to the standard error.

If you do not follow the assignment requirements -- especially if you deviate from them in a way that costs us time while marking -- you will lose points even if your scripts work correctly. So please read the directions carefully. This part of programming.

Possible Penalties:

- 10% if you hand in during the late period (24 hours after the deadline)
- 10% if there are administrative issues that cause the graders extra time