# Homework - 2

---

**Problem 1(3 points):**
For the first problem, you will get some practice with sed and grep by writing commands to parse data inside a dictionary file.  which is a newline-delimited list of English words. Each task in this problem-1 can be solved multiple different ways. In particular, many can be solved with either sed or grep, since there's a lot of overlap in what they do; for this problem, feel free to use either tool.
(Note: Do not use tools other than sed and grep)

To begin, use the command below to download the dictionary file from my github website:
curl -Lo dictionary.txt https://md685-njit.github.io/dictionary.txt

The above command should create a file named dictionary.txt, You can run "**cat dictionary.txt**" to see what the dictionary of words looks like (FYI, the file has 235885 lines).

Perform the following Tasks in a bash script:
1. find the number/count of words in dictionary.txt that have at least three a's (eg:  banana apparatus, madagascar, etc). (case insensitive)

2. Next, Find the number /count of words in the dictionary that have exactly three 'e's and where all 'e's are separated by at least one non-'e' character. In other words, you should match a word such as "serenely" (as all three 'e's have at least one non-'e' character separating them) but you should not match a word such as "beekeeper" (as there are two adjacent 'e's that are not separated by a non-'e' character). This is a little tricky as you need to consider that some words begin with an uppercase 'E' and regexes are case sensitive by default.

3. Finally, find the three most common last three letters for words in dictionary.txt that have two adjacent 'e's. I recommend doing this in two steps:

> 3a. Find a list of all words in dictionary.txt that have two adjacent 'e's. (The words can have more than two 'e's as long as at least two of them are adjacent).
>
> 3b. Then find the three most common final three letters of these words. (Hint: take a look at how you might use grep to strip off the final three letters and how you might use sort and uniq to find the most frequent occurrences for a given set of data).

**Problem - 2: (2 points)**
Write a Bash script that:
Searches for all lines containing the word “scanf” or “printf” in a given ‘C’ file passed as an argument (using grep).

1. Counts the number of lines containing “scanf” and “printf” separately.
2. Redirect the lines containing “scanf” to a new file named scanf_log.txt and the lines containing “printf” to a new file named printf_log.txt.
3. If the file scanf_log.txt or printf_log.txt already exists, the script should append the lines to the existing file instead of overwriting it.
4. The script should also print the total number of lines in the input file and the percentage of lines containing “scanf” and “printf”.

**Problem - 3: (2 points)**
Write a Bash script such that:

1. The script takes a single argument, which is the name of a directory. This directory should contain text files, each containing a list of email addresses. The email addresses can be spread across multiple lines within each file. In other words, there is only one email address per line.
2. the script should recursively traverse the directory and its subdirectories to find all text files. (Do not use find method)
3. For each text file, the script should extract all unique email addresses contained within it. Use regular expressions to identify valid email addresses. Valid email addresses should follow the format "username@domain.com". (eg: John@njit.edu, Kathy@google.com )
4. After extracting the email addresses from all the files, the script should combine them into a single list and remove any duplicates.
5. Finally, sort the list of unique email addresses in alphabetical order and save them to a new text file named "unique_emails.txt" in the same directory where the script is located.

**Problem - 4: (3 points)**
Write a Bash script where:

1. The script should first read a file line by line, where each line is a person's information formatted as "FirstName LastName, YYYY-MM-DD, City, Country".
2. The script should filter the lines to only include people who are from a city that contains two or more words (e.g., "San Francisco"). Use a regular expression to check this.
3. For each of the filtered lines, the script should extract the person's age in years, based on the date of birth. You'll need to use a regular expression to extract the date of birth and then calculate the age.
4. After all ages are calculated, sort the people by their ages in descending order and print each person's name along with their age.

Note: You can assume that the current year is 2024 for age calculations.