A big-data processing task:

Write a bash script to find out ***5*** most frequently used words on a set of Wikipedia pages. The script prints out a list of these words and the number of occurrences of each word on the Wikipedia pages. The list should be sorted in descending order based on the number of occurrences.

The following is a sample of output generated for 4 Wikipedia pages.

```
1080 the
677 of
480 in
473 and
443 a
```

Since there are a huge number of pages in Wikipedia, it is not realistic to analyze all of them in short time on one machine. In this problem, your script only needs to analyze all the pages for the Wikipedia entries with two capital letters. For example, the Wikipedia page for entry "AC" is *https://en.wikipedia.org/wiki/AC* . Thus, the pages we need to analyze are

https://en.wikipedia.org/wiki/AA
https://en.wikipedia.org/wiki/AB
https://en.wikipedia.org/wiki/AC
...
https://en.wikipedia.org/wiki/ZY
https://en.wikipedia.org/wiki/ZZ

Your script combines a few tools in Linux to finish the above big-data processing task. You can use wget to download and save a page. For example, the following command downloads and save the AC wiki page into file AC.html:

```
wget https://en.wikipedia.org/wiki/AC -O AC.html
```

A HTML page has HTML tags, which should be removed before the analysis. (Open a .html file using vi and a web browser, and you will find the differences.) You can use `lynx` to extract the text content into a text file. For example, the following command extract the content for entry "AC" into AC.txt

```
lynx -dump -nolist AC.html > AC.txt
```

After the contents for all the required entries have been extracted, you need to find all the words using `grep`. You need to use a regular expression to guide `grep` to do the search. All the words found by `grep` should be saved into the same file, which is then used to find the most frequently used words. Note that you need to find distinct words and count the number of times that each distinct word appears in file. Using the `-o` option (i.e., `grep -o`) will simplify the processing, since you only need the matching parts. You may need `sort`, `cut` and `uniq` in this step. Read the man pages of `sort`, `cut` and `uniq` to understand how this can be achieved.

Hint: You don't need to write code to count the number of occurrences for each distinct word. Use `sort` and `uniq` smartly --- sort groups the occurrences, and uniq counts the number of occurrences.