Submission Instructions:

1. For each problem, submit **1)** your script/program, and **2)** one or two screenshots in **jpg** format showing that your program really works.

2. Name your script and program using the pattern NJITID#_1.sh and NJITID#_2.py. Name your screenshots using the pattern NJITID#_Problem#_Index.jpg. NJITID# is the eight-digit NJIT ID (Not your UCID, Rutgers students also have NJIT IDs). Problem# is the problem number (e.g., 1, 2, 3, 4, etc). Index is the index number showing the order of the screenshots (e.g., 1 or 2).

3. Submit individual files. **DO NOT SUBMIT A ZIP FILE**.

A Big-Data Processing Task:

We need to find out ***15*** most frequently used words on a set of Wikipedia pages. Specifically, we need to find out a list of these words and the number of occurrences of each word on the 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
375 advertising
345 to
191 from
189 on
164 is
160 for
149 or
133 as
128 by
126 that
```

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. Here we only need 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

Problem 1:

Write ***a bash script,*** which 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.

Problem 2:

Write ***a python program*** to finish the above big-data processing task. Use `urllib` or `urllib2` module to download a page. Use `re` module to search for words.

A HTML page has HTML tags, which should be removed before the analysis. Use `BeautifulSoup` to convert a text from HTML format to text format.

Check the attached slides for how to use `urllib/urllib2` and `BeautifulSoup`.

Note: It is possible that the list generated by the python program and the list generated by the bash script differ slightly. For example, the word “Wikipedia” is included on one list, but not on the other; for a word on both lists, the count may be slightly larger on one list than the other. This is caused by the different tools used in bash and python to convert HTML into text. For the same HTML page, the text files generated by lynx and BeautifulSoup may be slightly different. But the results generated by your bash script and python program should not differ significantly. If you see the words on these two lists differ by more than 50% or the counts of the same word differ by more than 50%, probably there is a bug in you script/program.