Final Project (100 points)

Requirements

1. **NOTE: It is NOT a one-day project. Please start working on the project ASAP.**
2. You can form a team of two to work on the final project, or you can work individually.
3. Please submit the code as well as a report (see the report format).
4. Only one student in each team needs to submit the project report and code. Please include the names of both team members in the project report.
5. Grading: Report: 60%, Code Readability: 20%, Results: 20%

Report Format (use 12pt Times New Roman Font, at least 4 pages, at most double space, submit in PDF format)

Section I: Objective and Summary

- In this section, describe the objective of your project, and briefly summarize what your team has achieved in this project.

Section II: Overview and Detailed Design

- In this section, describe the overview of the achieved project, detail the design of each component of the project.
- Figures and flowcharts are encouraged in this section.
- Also discuss your designs/efforts to balance your code among run-time performance, memory overhead, and disk storage (if any).

Section III: Testing Procedures and Results

- In this section, describe the cases/scenarios you use to test your design, and also give the testing results.

Section IV: Conclusions

- In this section, summarize what you have achieved, highlight the advantages of your design as well as drawbacks/limitations.

Project Description:

In this project, you need to create an executable file **synonym (or synonym.py)** to get all synonyms of each given word.

Usage: **synonym** [**word**] [[word/optional] [word/optional] ….]

Examples:

- **synonym amazing:** awesome, fascinating, incredible, marvelous, prodigious, shocking, stunning, surprising, unbelievable, wonderful
- **synonym pass:** canyon, cut, gap, gorge, passage, passageway, path, ravine
- **synonym nosuchwordexists:** give an error message saying no synonym can be found.
- **synonym amazing pass:** first give all synonyms of amazing (i.e., awesome, fascinating, incredible, marvelous, prodigious, shocking, stunning, surprising, unbelievable,

wonderful), then given all synonyms of pass (i.e., canyon, cut, gap, gorge, passage, passageway, path, ravine)

Requirements:

- The synonym database is at https://www.thesaurus.com/. In particular, the website https://www.thesaurus.com/browse/[word] gives you the webpage content that shows all synonyms of [word].
- In order to reduce network traffic and improve efficiency, once your code obtains all synonyms of a given word, it should store them at local storage for the next use. In other words, your code, upon start, should first check whether its local storage has stored all synonyms of the given word. If yes, just return the locally stored information; otherwise, get the synonyms from thesaurus then store them. Carefully think about what kind of data you should store and how to store such data.
- The website thesaurus can periodically update its database. Thus, the storage of synonyms for a given word should be associated with an expiring time (you can set this time to a value that you feel reasonable), your code should check whether the storage time has expired. If yes, your code should re-obtain the information from thesaurus, and then update the local storage (with the new expiring time).
- Create a batch-run Linux script, which saves all obtained the synonyms of all words given by an input file to an output file. Please create your own input file with at least 30 words. Your script should first read the words from the input file, then repeatedly run **synonym (or synonym.py)** for each word and obtain the corresponding result. All the results are finally saved by the script to the output file.

Tips:

- You can use Google to get familiar with popular Python libraries that can get webpage content. Carefully go through the code examples.
  - common web Python libraries: requests, urllib
  - On Nov 18, a code example will be given and discussed.
- Be aware that a website might not allow aggressive requesting of its webpages within a short time duration. In addition, always obtaining webpage contents from the Internet may slow your design and testing. Therefore, when designing and testing your code, try to minimize the frequency of getting webpage content from the Internet. Generally, you can store a few web contents into local files first, and then write/test the majority part of your code based on these files. At the final step, you can add the webpage request part into your code to do the integrated tests.
  - The source file of a web page can be seen if you select an option like ‘view page source’ in the right-click menu inside a web browser
  - You may need to manually analyze the content of a web page to find a way (e.g., using regex) to extract the synonym information you need.
- If somehow the webpage request part in your code does not work (there is just a slight penalty if this part does not work), you can manually store some webpage content files for a few words and use them for your regular expression based string processing as well as the entire code testing and evaluation. Submit these manually stored webpage content files along with your final code.