CSC 351L – Comparative Programming Languages Lab
Project 5 – Prolog Concordance
Due: March 18, 2021 at 11:59 pm

Description:

A concordance is a listing of the words that appear in a large document along with the numbers of the lines or pages where those words appear. A concordance is useful to researchers of texts who want to explore how various words are used in those works.

For this project, you will create a concordance for a text file. To do this you will need to read words from a file, inserting them into an alphabetized list of words, and making note of the line numbers where that word appears. After the entire file is read, you should print out the words in alphabetical order, along with the lines where that word appears in numerical order. If a word appears more than once on a line, the line number should be printed once, with the count printed in parenthesis after it.

For example, a piece of a concordance might look like the following:

```
apple 1, 4, 6(2), 12
book 2, 5, 7
cabbage 3, 6, 9
```

In this example, "apple" appears twice on line 6, where all of the other cases appear once on each of the lines indicated. Note: there should be commas after each line number except for the last.

You can assert facts to keep track of the lines where the words appear. Your facts should have the form `appears(apple, 6, 2).` to indicate that `apple` appears twice on line 6.

You will be provided with the predicates that read one line of the file and make it into a list of atoms. The version of the input for this project is different than the one in project 4 because it needs to keep track of the lines and doesn't stop at the end of a sentence. The provided code will keep track of the current line number by asserting the fact `line_number(X).` where `X` is the actual value.

Steps:

1) Copy the program file found at:

   ```
   ~mcconnel/CSC351/Prolog/Project5/concord.pl
   ```

   This file includes the main loop, updated reading predicates that handle end of line and the line number, a reset so the program can be run more than once without exiting Prolog, and a set of predicates that alphabetically compare atoms. You are also given predicates that skip the end of line so it's not treated as another word. The `concord` predicate handles when the end of the file has been reached.

2) Write the `process` predicate that will handle a word by first checking to see if it already appeared on this line and if it did, updating the count.
   - This might be easiest if you have a separate predicate that returns the current line count or 0 if the word has not yet been on this line.
   - You should realize that if you use `assertz` for the new counts, the line numbers will always be in order.
3) You will need to assert a fact `my_word_list(X)` where `X` is the alphabetical list of the words found in the file so far, with no duplicates and no line numbers. This list will be used to output the words in alphabetical order.
   - You will need a predicate that creates a new list with the next word "inserted."
   - You will need a second predicate that retracts the old list, calls the predicate that "adds" the new word, and then asserts the "new" list it creates.
4) You will need a set of predicates (I used three) to handle the printing.
   - My first goes through the alphabetical list printing the word and calling the second predicate to handle the line numbers.
   - The second one gets a line number and count and lets the third predicate decide whether to just print the line number or the line number and the count.
   - Your printing should include commas only where necessary.

Deliverables:

When you complete the program, you will prepare two things: a project report and a zip file containing all of your code.

The project report must use the format given in the sample file on D2L. This report can be prepared as a text file, MS Word document, or PDF. The project report should not be included in the zip file.

The zip file should include all prolog files. You can create the zip file in one of two ways – using zip on brahe or on your own computer.

The two items must be uploaded to the Project 5 drop box on D2L. No other form of submission will be accepted.