Bonus assignment week 7–8: a better word processor

Course 'Imperative Programming' (IPC031)

1 Assignment

The naïve version of `read_word` that uses `>>` in the mandatory assignment may consider strange character combinations as a single word. For instance, in the novel "`Alice's Adventures In Wonderland.txt`" the red-underlined parts should not belong to a word, but the following entries are accepted as words:

- "Wonderland,"
- "whatsoever."
- "think--'"
- "(for,"
- "distance--but"
- "Australia?'"

It is a known hard problem to invent a solution that works for all conceivable texts. For this reason, we restrict the definition of words as follows:

- A *word* is a sequence of one or more occurrences of a *letter-character*, followed by zero or more occurrences of a *connector*.
- A *connector* is a single hyphen '-' followed by one or more *letter-characters*.
- A *letter-character* is:
  1. one of the lower-case letters 'a' up to and including 'z'
  2. one of the upper-case letters 'A' up to and including 'Z'
  3. the apostrophe "'"
- A *garbage-character* is any character that is not a letter-character.

Examples. According to this definition, "waistcoat", "rabbit-hole", and "one-two-threes" are individual words, but "distance--but" has to be read as two words. The "`assignment-07-bonus-files.zip`" file on Brightspace contains an additional text file "`hyphen_madness.txt`" to test your function.

Part 1: Implementation

Take your finished "`main.cpp`" from the mandatory assignment as your starting point. Design and implement an improved version of `read_word` that reads words following the above definition of word, which replaces your old version of `read_word`. You should not have to make any other changes in your code, though you are allowed to introduce additional helper functions to help implement `read_word` if desired.

Part 2: Testing

The testing to perform is very similar to the mandatory assignment. In particular you must:

- Implement the `enter_command` test cases, which now also includes a test case for the test text file "`hyphen_madness.txt`".
- Ensure the provided tests for `count_command` work and pass. Note that the expected number of matches, especially in the Alice text, may differ compared to the mandatory assignment due to our new definition of words.
- Perform manual testing of the `content`, `stop`, `where`, and `context` commands to validate your program as a whole still works as intended with the new definition of words.

2 Products

As product-to-deliver you upload to Brightspace:

- "`main.cpp`" that you have adopted to work with our new definition of words.
- "`main_test.cpp`" that has been extended with non-trivial unit tests for each new function that you have developed in "`main.cpp`", along with your test implementations for `enter_command`.

Deadline

Bonus assignment: Friday November 8, 2024, 23:59h

Important notes:

1. check that you have actually submitted your solution in Brightspace.
2. the deadline is firm, and it is impossible to upload a solution after expiry. In that case you fail the assignment.
3. you can upload solutions to Brightspace several times, but only the last submission is stored.
4. identify yourself and your lab partner in every uploaded document. The identification consists of your first and last names, student numbers, and number of (sub) assignment. By identifying yourself, you declare that the uploaded work has been created solely by you and your lab partner.
5. your work will be judged and commented upon. We expect that you obtain the feedback, read it, and use it to for the next exercises.
6. it is essential that you only submit your own solution, never copy somebody/something else's solution, and never share your solution—in particular: **AI tools (including but not limited to Github Copilot or ChatGPT) are not permitted**, solutions from previous year cannot be reused, and finally, you and your lab partner take joint responsibility for the assignment you submit.