CSE 303, Spring 2005, Assignment 5C (I/O)
Due: Tuesday 17 May, 9:00AM

Last updated: May 16

Summary: You will write a C file that implements an interface for returning successive words that are read from a file. You will also write unit tests for the code you write. The sample solution, not including unit tests, is about 55 lines.

You may *not* use any global variables; any state that your file needs to maintain in order to return the next word must be pointed to by the "info pointer" described below. For actually reading input from the file, use the `getline` function as in previous homeworks.

1. Include the following type definitions:

   ```
   typedef struct InputInfo * input_info_t;
   struct InputInfo {
     ...
   };
   ```

   The fields of `struct InputInfo` should include:

   - A `FILE*` for the file from which input is read.
   - A `char*` for the buffer whose address is passed to `getline`.
   - A `int` describing the length of the buffer passed to `getline`. (This field's address is also passed to `getline`.)
   - A `char*` that points to the "current position" in the buffer.

2. Implement the function `input_info_t initialize_input(char * filename)`. It should open the file named `filename` for reading (printing an error message and exiting the program if opening it fails). It should return a pointer to a new heap-allocated `struct InputInfo`. It should read the first line of the file and initialize the fields of the `struct InputInfo` appropriately. In particular, the "current position" should be the beginning of the buffer holding the first line of the file because no words have been read yet. You may assume the file has at least one character in it.

3. Implement the function `char * next_word(input_info_t info)`. The result is a new heap-allocated, `'\0'` terminated string containing (a copy of) the next word in the input file stored in a field of `*info`. Words contain only English letters (either case) and the hyphen (`'-'`) character. Any other character is not part of a word. Words are separated by one or more other characters. If there are no more words in the file, return `NULL`.

   Notes:

   - The standard C library has a function `isalpha`.
   - Some executions of `next_word` will need to call `getline`, but others will just advance the "current position".
   - It is up to the caller to free the space for the string when it is no longer needed.

4. Implement the function `void complete_input(input_info_t info)` to close the file and deallocate the space consumed by `*info` and the buffer returned from `getline`. Be sure you close the file, do not create space leaks, and do not follow dangling pointers. It is up to the caller not to use an `input_info_t` after passing it to `complete_input`.

5. Write *unit tests* for the functions above. Include comments indicating what different tests accomplish and a `main` function that runs your tests.

Extra Credit: Change your implementation such that hyphenated words can span lines. That is, if the last character on a line before the `'\n'` character is `'-'`, then the word continues onto the next line. Note that a word could span any number of lines, not just 1 or 2.

Assessment: Your solutions should be:

- Correct C programs that compile without warnings using `gcc -Wall`.
- In good style, including indentation and line breaks
- Of reasonable size

Turn-in Instructions:

- Follow the link on the course website and follow the instructions there.
- **Problems 1–4 should have solutions in `word_io.h` and `word_io.c`; your unit tests should be in `word_io_test.c`.**
- We should be able to compile your program via `gcc word_io.c word_io_test.c`, assuming `word_io.h` is in the same directory.
- You can also turn in one test input file.