Assignment 1: Textbuffer FAQ

general questions

Can I modify `textbuffer.h`?

No.

Can I add my own `structs` and functions to `textbuffer.c`?
Yes! Make sure your functions are declared `static`, and you document what the functions and structures you add are for.

Can I use functions from `<string.h>`?
Yes. It's much, much harder if you don't.

Will I need to check my code for memory safety or memory leaks?
Yes. We certainly will.

Can `Textbuffer` be defined like `link` in the lecture examples?
If `Textbuffer` points directly to the head of the list, functions like `textbuffer_merge` cannot function correctly, as they may change the head of the list.

If I'm about to *abort(3)*, should I *free(3)* the textbuffer?
It's a bad idea to clean up after yourself before *abort(3)*'ing if you have bad inputs, just in case the reason you got bad inputs spreads.

`textbuffer_new`

How does `textbuffer_new` work?
If the input text is, for example, `"Hi\nhow\nare\nthings\n\0"`, the buffer should contain the following lines: `{ "Hi", "how", "are", "things" }`. You will have to process the input string, extract all the substrings separated by newline, and copy them into the entries of your buffer structure.

Should I leave the `'\n'` characters in?
Depending on your approach to splitting text, they may already be The only other place you need the `'\n'` characters is in `textbuffer_to_str`, so you could probably get away without storing them.

Is it safe to assume that the text will always have a newline at the end?
Yes, text will always have a newline.

What should happen with multiple consecutive newlines?
Each newline marks a new node in the text buffer. You need to track empty lines.

Can I use *strtok(3)* or *strsep(3)*?
I recommend *strsep(3)*, but you can use either, though you should be careful about using *strtok(3)*. Note, however, that to use either, the input string needs to be mutable… and this isn't guaranteed in the spec.

`textbuffer_drop`

How should I write tests for `textbuffer_drop`?
You cannot. You can't write a black-box test for a destructor.
When you `free(3)` memory, all you're saying is that you no longer need the block of memory you had a pointer to; it should be irrelevant to you whether that memory's value changes or becomes invalid in some way, because *you are absolutely forbidden from accessing the memory once free'd*. Use after free is an illegal and undefined operation. You have no way to invalidate the pointers (read: change any values outside your ADT, including outside pointer references to its state structure).

A good test that your `textbuffer_drop` worked is that your program is still running after you do so.

`textbuffer_to_str`

My textbuffer has no lines; what should `textbuffer_to_str` return?
The empty string. (Our tests will also accept `NULL`.)

`textbuffer_swap`

Should I swap the string pointers or the whole nodes?
You should swap **nodes**, not the string pointers.
If somewhere in your textbuffer, you've got pointers to a particular node, and that node's *value* changes, that node's identity is lost. With the exception of `textbuffer_replace`, each node's value (in essence, the line of text it holds) should never, ever change.

`textbuffer_insert`

What should happen if I `textbuffer_insert (tb1, 1, tb1)`?
Attempts to merge a textbuffer with itself should be ignored.

Should I call `textbuffer_drop` as well?
No!

Can I concatenate text buffers with `textbuffer_insert`?
The correct behaviour should be as follows, for `textbuffer_insert (dest, pos, src)`:

- `pos = 0`: insert `src` before the start of `dest`.
- `pos = textbuffer_lines(dest) - 1`: insert `src` before the last line of `dest`.
- `pos = textbuffer_lines(dest)`: append `src` at the end of `dest`.

`textbuffer_cut`

What happens if I cut the whole textbuffer?
You have an empty textbuffer… and you give back a copy of the new textbuffer.

`textbuffer_replace`

Will `str1` and `str2` be the same size?
Not necessarily! You might be replacing one string with a longer string, and your original chunk of memory may not be big enough. You will need to solve this problem.

Should I search or replace the entire line or a substring?
You should search for, and replace, substrings.

Will the search or replacement strings include newlines?
No.

Are the search and replacement strings case-sensitive?
Yes.

What happens if I search for the empty string, `""`?
Nothing is done in this situation. Nothing should be replaced.

What happens if I replace with the empty string?
Occurrences of the search string are removed.

`textbuffer_diff`

Does `textbuffer_diff` change either of its textbuffer arguments?
No. `textbuffer_diff` is non-destructive.

`textbuffer_undo` and `textbuffer_redo`

What is the effect of an undo of `textbuffer_insert`?
You don't have to re-create the original textbuffer; you have no way to give it back, anyway. Just reverse the effects of adding the new lines to the textbuffer.