COSC 2P91 – Assignment 2

Due: Mon Feb 23 @ 11:59pm

For this assignment, your task will simply be the creation of a sorted (or sorting) dynamically allocated structure.

Background:
This is going to be a basic text-processing task. Your program will be receiving numbers (integers) from *standard input*, and storing them in some data structure. You may not make any assumptions about the number of values a priori. As such, a static structure (like an array) is *not* permitted!* Values will be newline-delimited (that is, simply one number per line). Duplicates may not coexist within the structure (they are simply ignored if they're already in it).

When the user enters the number zero (0), all numbers that have been buffered thus far will be printed to *standard output*, but in (ascending) sorted order. Each value will be printed on a separate line. No additional whitespace will be added. As values are printed, they are also removed from the buffer.

If the user enters zero when there are no values buffered, that zero is simply ignored.

Of course, zero – being a control character – will never be added to the list (or included with output).

When the program encounters EOF (end-of-file), it will behave as a zero if there were any remaining buffered values (printing them out in ascending sorted sequence) and then close the program. No other output is permitted.

Requirements:

- Your program must behave as described above
- Split your program into (at least) two source files:
  - One for the main program and handling reading/writing
    - `program.c`
  - One for handling management of the data structure
    - `storage.c`
    - You don't need to write this as a library, but still give reasonable consideration to design
- Use `structs` to create your dynamic storage
  - My suggestion would be a sorted sequentially linked list, but a Binary Search Tree would also work just fine
- Ensure that you include appropriate memory management
  - Yes, the marker will notice a memory leak
- You may use the `stdio`, `stdlib`, and `string` libraries
  - Feel free to ask if you want to use something else

* I say you aren't allowed arrays only because you must make a dynamic data structure. However, if you should require arrays for intermediate structures (e.g. to hold a string, or to facilitate sorting), then that's acceptable. Definitely make sure to include enough commenting, though, so the marker can easily tell what everything's for!

Tips:

- Don't worry too much about the standard input and standard output stuff. The `scanf` and `printf` functions use standard input and output, respectively. Of course, this means your program can be piped into itself...
  - To ensure you understand how the program should work, consider what you'd expect the output to be like if you were indeed to try piping one instance of the program into another
- `scanf` will return `EOF` if it hits the end-of-file
  - `EOF` is just a macro-defined integer
- End-of-file might not initially seem intuitive to be dealing with, but it is also used for a de facto end-of-stream. Thus, if you hit *ctrl+d*, that will end the EOF, indicating that you're done typing things in
  - Incidentally, that's also a quick way to close a terminal when logged-in
- I've already said this, but definitely be careful with memory management. Consider using `top` to monitor your program as memory is allocated and (presumably) deallocated
- Though duplicates may not coexist within the structure, no consideration is given to history after the structure is dumped
  - e.g. 5 → 0 → 5 → EOF will yield 5 and then 5 again
- File names listed above are mandatory

Submission:

Both physical submission and electronic submission are required.
For physical submission, print out all of your source code as well as a sample execution or two, staple everything together along with a departmental cover page, and submit it in the dropbox. If necessary, you can use an envelope, but that really shouldn't be necessary.
For electronic submission, put your source files within the same folder on sandcastle, enter that folder, and run `submit2p91`.