Adit Patel

README:

Tokenizer reads an input, and produces tokens out of them.

I first create the token, using argv[1] in the main function. In create, i have a string, a length, and an index. All of these help me get and print out the appropriate output. The index is updated in my getNextToken method. After creating the token in main, i make a new string and call my method TKGetNextToken.

In here, it basically splits up the initial input string by spaces. That is the first thing that happens when it goes into the function. I do this by getting the first non-white character, and then getting the next white space character. If I get a char that is not a digit, i print out invalid, and then the hex for it. In between these 2 indexes, will be my string which I then put into the switch table to see which one it is. I used memcopy to copy the chars in between the indexes. I also had the manually set the nullbyte, if I wanted to do it this way because the output was supposed to have the exact right amount of bits.

For my switch table, i started at 1 because of the whitespace in the beginning. In this switch, i had zero, octal, decimal, hexadecimal, floattype, mal, floatEtype, floatSigntype. I thought it was intresting that I followed the fsm, but not explicitly. I kind of started backwards on it instead of fowards. In the switch i had the end types. From here, I used the path(s) to get to the end type. I had to make multiple endtypes for floats since there were so many options for it.

All of that is in my getNextToken function. Back into my main function, I had a while loop so it did it for all the tokens until there were none left. From the floatEtype and floatSigntype I had previously used in order to easily trace the path, I again changed the type of these to to just floattype since they are floats. If the current char* was a mal, i outputted the hex form of it. Otherwise, I printed the type and the token using an array of strings that I made. This array had all the types, and I knew which one to use by using the enum. After this, I freed the string so that it resets, and then called getNextToken again in order to get the next token. This repeats until the string is empty.

After it ended, I called tkDestroy. In tkDestroy i essentially just freed the str in the struct and tk itself, and also set them to null.