Tokenizer

Seong Kon Kim

(1) **TokenizerT Type**

1. count- going through the string and counting the characters
2. ptr-pointer to get the remaining string
3. original- argv[1]
4. statetoken- stores the Tokentype of the string
5. ec- equals 1 if there is error. Vice versa
6. error- the location of the error

(2) **NewToken (Function)**- changing the variables to their initial value.

(3) **TKGetNextToken (Function)**

As the program goes through this function:

1. It looks at the white space, just in case the string starts with them. It deletes the whitespace and put the new string into count.
2. The function follows the Finite State Machine. I divided mal Function and invalid differently. Just like how the FSM said, for example, if the input is “0.”, then that is mal Function. Also if the input is “0.e” then I output as “0.” as mal Function and ‘e’ as a invalid value and put it as hex number.
3. I also used #define so that it is easier to find where the string is passing in the FSM.

(4) **Error (Function)**- using this function, if there is an error in the string it will print out the error in hex number.

(5) **main (Function)**- I expect that there will be an empty string so I used argc to avoid the segment default. Else, I put a loop for all the functions until my pointer array gets empty. And as the program terminates, the tokenizer will destroy a Tokenizer objects.