Jessie George
RUID 157000770
Project 1

Special features:

1. Print message for unfinished tokens like “0x”
2. Print an error when user doesn’t enter anything for argv[1]
3. Print message when user enters empty token stream “”
4. Print message when user enters token stream which contains only white space or null characters
5. To increase efficiency, if length of token stream is 1 and the character is not a digit, print an error. (This avoids going through the whole switch case to find the error).

Assumptions I’ve made based on question:

1. “0” is considered an octal
2. No negative decimals like “-34” based on the Professor’s Finite State Automaton. So output will print error hex value of ‘-‘ and then print decimal 34
3. A float can contain either a decimal point or an exponent or both. So for example “1.8” and “10e-1” and “3.4e-1.4” are all valid floats
4. Any number starting with 0 cannot be a decimal. So if input is “089”, it cannot be decimal, octal or hex, therefore output is “unfinished token” because the number can become a float.
5. TKGetNextToken returns null instead of 0 to avoid ambiguities when input is “0”.
6. TKGetNextToken does not give null delimiters to increase efficiency (or else it is a wasteful call of the append function because the switch case is built to ignore null delimiters based on the Professor’s announcement).
7. Escape character is anything which is not part of the token or whitespace, read in one bit at a time because of the single-pass rule.

Logic and control flow:
Accept token stream through argv[1]. Create an object in TKCreate. Loop through the entire stream of tokens while it is not null. Call function CheckTokenType to check if a token is decimal/octal/hex/float/error. When we reach a character we make a series of decisions to eliminate impossible options (example: ‘X’ in the second spot means everything but hex is impossible).
We use an int array called impossible to keep track of states where value 1 = yes (state is impossible) and value 0 = no (state is possible). Note:
impossible[0] = whether a decimal is possible or not
impossible[1] = whether an octal is possible or not
impossible[2] = whether a hex is possible or not
impossible[3] = whether a float is possible or not
I coded helper functions like Digit, Alphabet and DecimalPoint which identifies the state of the current character and makes decisions about the type of token.
When we reach a delimiter i.e. whitespace, null character or escape character, we call TKGetNextToken which gives us the new token stream and also reinitializes the counters.
Other helper function include printTok (to print the token) and printErr (to print the error).
Once we have finished all the output, TKDestroy frees the space.

Note: In my program I put TKGetNextToken above TKCreate just so I wouldn’t getting a warning for implicit call when I compiled but I haven’t changed any of the framework which Professor Russell gave us.