Amar Bakir
CS 214
Project Assignment 1: Tokenizer
Wednesday, September 17th, 2014

The way my program works is it accepts two strings from the command line, the first being the string of separators and the second the string to be tokenized. First it runs the strings through the function PreProcessString that is built to specifically remove rogue backslashes at the end of the string to be tokenized and to process both strings for special characters. This is done in accordance with the requirements for the assignment (hopefully) in order to make the tokenizing job much easier. There is an int that this function takes. All that does is tell the function whether or not to remove backslashes at the end of the string it is handling, since for the separators I do not want to just remove all backslashes while for token string I do.

After that the main method runs through a loop that parses over the now processed string. Whenever it hits a non-separator character, it calls the tokenizing function to start tokenizing from the start of the non-separator character. The tokenizer object holds a huge array to serve as a buffer and an int to hold the length of the current token in the buffer. The tokenizer function TKCreate takes the strings and runs through the string to be tokenized, copying any non-separating characters to the tokenizer object's array, until it finds a separator character. It then null terminates the string it already has and then the tokenizer object returned from TKCreate is passed to the function responsible for grabbing the token as a char string.

All the TKGetNextToken does is copy the token over char by char, while allocating memory for it using the token's length (stored in the token struct) so it knows how big the char string array needs to be. That char string is then printed by passing it to the function TKPrint which handles the special characters and prints them out as specified in the assignment (using [] with the hex value inside it). All used memory is then freed, and the loop goes on to the next part of the string. The loop counter is moved up by the length of the found token so as not to repeatedly print the same token out, and the process is repeated until the loop hits a null terminating character or runs for the length of the string.