The primary data structures used in the creation of this tokenizer program were char pointers that contained separator and token arrays. The manipulation of these pointers allowed me to print out the appropriate token strings without designated separator characters.

The TKCreate function involved initializing and properly allocating memory space to the char pointers. I also had a created_token variable in my struct that was responsible for maintaining the tokenized string from the TKGetNextToken function. The counter variable was responsible for maintaining the appropriate starting place of the token string on each iteration.

TKDestroy is responsible for freeing all the memory allocated space in this program. The three pointers used in this assignment were freed upon completion of the string stream. Created_token was also freed once the string was completely read.

TKGetNextToken is responsible for returning a tokenized string that has been delimited by separator characters. However, escape characters were left in the string, and they were handled in the main function in terms of hex modification and separation.

The main function handled the creation of a TokenizerT object, and the iteration of the token stream with each separator character. Escape characters were handled individually, and the application of hex address printing was implemented. Separation of escape characters was also handled in the printing process of the token.