Main Method: My programs starts off with checking if there's input from the command line. If there's no input, the programs terminates. If there's input it allocates memory and calls the TKCreate function and create a token with the char **list variable and has the token point to list. It tokenizers input and after it's done, it frees the list and token variables.

Int fltCheck/decCheck/octCheck/hexCheck: These functions check if the char array is a float, hex, octal, or a decimals. Returns a 1 if true and 0 if it's false.

Void combine: This functions takes a char c and adds it at the end of a char array s. It makes the last element in the array c, and the element after c is the null terminated '\0'

Void check: This function takes a char array. It prints the array, then checks if it's a float, hex, octal, or decimal using the four functions listed above. It prints whatever ended up True. If all return false, "MAL" is printed. The memory for the char array is reset at the end.

Void tokenizer: This functions goes through the string one time. It checks one char at a time using a loop. If the current char is a hexadecimal, x, X, . , +, or -, it uses the combine method to at the char c to the end of the char array temp. When there's a space, check method is called for the char array temp. If there are multiple spaces, the loops goes to the next char, and does not call check. If there's an escape char, the array temp calls check only if the previous char wasn't a space, and then the escape char is printed out.

General Info: My program only checks if the token array is a float, hex, octal, or a decimal if there's white space or an escape char. So for a string "070032439" it would insert the char one by one into a char array and at the end it would call check. There are no spaces or escape characters in this string .The Token would be "070032439" and would be considered mal. For the token "123v213)d" The tokens would be "123" decimal, "v" [0x76], "0700" octal, and "z" [0x7a] . "z" and "v" would be the escape characters in this string. My code is efficient since it goes through the string once. The big O for my code would be n + 4kx where n is the length of the input char array and k would be the number of times the check method is called, since it goes through the token four times. X would be the average length of the token that is being checked.