The tokenizer will, given a string input, print each token on a separate line, identifying both the token's type and text in the format: ***TokenType: TokenText***. If there are any escape sequences entered (*/a*, */x02*, etc) into the string as their integer value in the string's character sequence (ie: if character *i* in the char array is of decimal value 2, it is considered an *escape character*), it will be printed on its own in the format: ***[hexValue]***.

The program attempts to make guesses as to what the type of the token is, and then verifies its guess by checking the entire sequence against the definitions of each token type. In effect, the token's first and, if applicable, second characters are evaluated to determine what type of token it *could* be. If token[0] is 0, then we can assume that the token is either of type *Hexadecimal*, *Float*, or *Octal*. If the second character is x/X, we can be fairly certain that the token is a *Hexadecimal*, and therein verify our result. If the second character is a decimal, then we can be fairly certain that the token is a *Float*, and therein verify our result. If neither of these conditions are met, we can be certain that the token is *Octal*, and therein verify our result. If we perform a verification and find that the token isn't what we expected, we return *MALFORMED*. Similarly, if the first character is a non zero character, we can deduce that we are either dealing with a *Decimal* or *Float*, and check for each, respectively (if a number is in the format of 1xxxx and it isn't a decimal, then it must be a float). If both of these checks fail we return *MALFORMED*.

The program uses a verification model for purposes of maintainability, as adding more types is as simple as adding additional verifications. Modifying existing definitions of a type doesn't interfere with other types. We attempt to guess which type it is before verifying to save computations - bringing the **overall running time of the program closer to O(N)**, rather than O(4N) (~O(N)) if every typed is checked sequentially.