1. Is there any duplication in the testing code you've written to test individual token regular expressions and the code used by scan? (Don’t have to fix)

We don’t have to fix this issue because we had already minimized the use of function scan to 1 per unit testing case for the regular expression of each token in scanner.h and scanner.cpp.

2. Does your scan function make calls to makeRegex every time it is called? (Don’t have to fix)

We did not put makeRegex in scan, but instead in the constructor of Scanner. With this implementation, makeRegex runs only once when a new scanner is made so it effectively improves the running time of the program.

3. Do you have a redundant array of tokenType values? (Don’t have to fix)

We don’t have to fix this issue because we never had these in our original iteration 1 code.

4. Would changing the order of intKwd and printKwd in the definition of enum tokenEnumType in scanner.h have any adverse effect on the rest of your code? (Don’t have to fix)

This wasn't an issue in our original code, but this question let us to make sure that nothing was more dependent on the enumType order.

5. Do you create a named regex_t pointer for each regex instead of only putting them in an array? (Daniel has already fixed this issues in Iteration1)

We actually discussed this issue during our iteration1 design stage, and Daniel made an important point that creating regex_t pointers not only takes up unnecessary memory usages in our program but it also makes our code become longer and more difficult for others to read, so we decide to create an array which contains 41 regex_t and each one of them is corresponding to one of enumtype tokens, so that we can iterate through all the key word easily through the indexes.

6. Are there any places in which enumerated tokenType values should be used instead of integer literals? (Don’t have to fix)

We have never coded any integers when referring to tokenTypes.