Question: I just wanted to clarify, for the LISTOP1_T variables (car, cddr, etc.) they need to have a space in between them in order to be considered a LISTOP1_T token right? Based on the example input and output given in P1-2.pl460 and P1-2.p1, it seems that when they are connected together (carcdr...), they are considered an IDENT_T token so I am pretty sure this is accurate but, I just wanted to be sure.

Answer: Yes. To be identified as a LISTOP_T, the r may not be immediately followed by another letter, a digit, an underscore, or a question mark. caddr+cdr is a LISTOP_T followed by a PLUS_T and another LISTOP_T. caddr1car is an IDENT_T.

Question: Do we use 'getline' from our 'getToken()' function to get each line?

Answer: Yes. To do this, you should create a loop to bypass whitespace at the beginning of your GetToken function.

Question: Regarding the constructor that takes in the "fileNamePrefix" as a parameter. Are we supposed to be passing it the prefix (Lex tokenizer("P1-1");) and opening the file from our constructor?

Answer: Take a look at how the constructor for LexicalAnalyzer is called in SyntacticAnalyzer.cpp and how the constructor for SyntacticAnalyzer is called in Project1.cpp.

Question: What is the purpose of the function: string GetTokenName (token_type token) If I gave this the token CONS_T, would it give back the string "CONS_T" or would it give back "key words"?

Answer: This function should return the string version of the name assigned to the element of the enumerated type. GetTokenName (CONS_T) should return "CONS_T".

Question: Does this language we're writing the lexer for need to have a space between identifiers? For example, is newlinecond two separate tokens or a user defined identifier:

Answer: newlinecond is a single user defined identifier. newline+cond would be a NEWLINE_T followed by a PLUS_T followed by a COND_T.

Question: Is 1+2 acceptable for the language? Or, does this need to be 1 + 2

Answer: Both are acceptable and would result in a NUMLIT_T followed by a PLUS_T followed by a NUMLIT_T.

Question: Is 1++2 the same as 1 + +2 or is it an error?
Answer: Yes, since our language does not include a ++ operator, 1 ++2 is equivalent to 1 + +2

Question: Can we change the numbering of the enumerating?

Answer: Yes, you may modify the token_type enumerated type as you need. All new elements of the type should be added before MAX_TOKENS. LISTOP2_T needs to be changed to CONS_T.

Question: What is IDKEY_T in our case? Do we first get IDKEY_T and from there check in our map if that specific lexeme has a different token type and if not, set it to IDENT_T?

Answer: IDKEY_T is a token that will be used internally. Once a lexeme is identified, a more specific token should be returned by the function. A map is probably the most efficient way to correlate a key word lexeme with its specific token. For example, display and DISPLAY_T.

Question: They’re error_BU and error_NBU, so then what’s error_T?

Answer: Like IDKEY_T, ERROR_BU and ERROR_NBU should be used internally. If either type of error is found the ERROR_T token should be returned by your GetToken function.

Question: Can I have a vector of chars so that getting the column for that char, to use for accessing my table? If so How would I treat the ‘other’ category?

Answer: You will need to develop a technique for associating specific characters with columns in your state table. In the RE1/RE2 example, there is a function called GetColumn that does this task. This is a simple function – but not an efficient way to solve the problem if there are a lot of possible characters in the alphabet associated with the regular expression.

```c
int getColumn (char one)
{
    if (isspace (one))
        return 0;
    if (one == 'a')
        return 1;
    if (one == 'b')
        return 2;
    if (one == 'c')
        return 3;
    if (one == 'x')
        return 4;
    if (one == 'y')
        return 5;
    if (one == 'z')
```
return 6;
  return 7;
}

**Question:** Functionality of reportError function

**Answer:** The ReportError function should write a message containing the line number, position on the line, and error message passed to it to the .lst file.

**Question:** Question

**Answer:** Answer