1. What is lexical analysis and how is it related to other parts in compilation?
Ans : The main theory of lexical analysis is to read each and every symbols that a code contains. For example, a = (b + 1) - c; Here 9 tokens have been generated where (and) have resaved tokens. In the next phase called syntactic analysis, a derivation tree is generated based on tokens that was created on lexical analysis. If the tree is not generated then compilation process is aborted.

2. How is the lexical structure of the language expressed in the PLY tool?
Ans: Ply consists of two modules one is lex.py which break input text into a collection of tokens and yacc.py recognizes language syntax. lex.py provides an external interface token() function which returns next valid token input stream and the repeat call invoked on ycc.py to retrieve tokens and get grammar rules. ycc.py outputs are Abstract Syntax Tree (AST). Ply depends on levers and parsers. No extra source of file and no compiler instruction.Result of ply si cached and stored in a file. Tables are read from cache, if not available then it generated again.

3. Explain how the following are recognized and handled in your code:

1. Keywords:
    - Keywords are stored in a dictionary and then added to token list as a list.
2. Comments
    - Regex for comments is r’\.\.\..*\.\.\.’ If found

then passed it by not considering as a token.

3. Whitespace between tokens
    - I used two kind one you can find on def t_WHITESPACE and another is t_ignore = ‘ \t’ removing one produces same result.
4. Operators & delimiters (<-, parenthesis, etc.)
    - I defined them on a single line. E.g. - t_COLON: r’\:’
5. Decimal literals
    - Regex for decimal literals is r’[-]?[0-9]+[.][0-9]’
6. Function names
    - Regex for function is r’[A-Z][a-z_0-9]+’

4. How can the lexer distinguish between the following lexical elements:

1. Function name identifiers & variable name identifiers
    - I followed given rules function name identifiers starts with capital letter and variable starts with lower case letter.
2. Keywords & variable name identifiers
    - Keywords are reserved in list. After detect variable that list is checked if true then keyword else variable.
3. Operators > (greater than) & >= (greater or equal)
    - by putting >= on the top on > because ply set priority from top to bottom.
4. Info string literals & variables names
    - Different in regex and put info string regex before variable.
5. Comments & other code
    - By putting comments regex on top of all.

6. Integer literals & a decimal literals
    - Different regex and put decimal before integer regex.

5. Did you implement any extras? If so explain them (what and how)
Ans: No I followed **SheetScript only.**

6. What did you think of this assignment? What was difficult? What was easy? Did you learn anything useful?
Ans: At first I was afraid and thought that it will be really difficult but by the guidance of my course teacher and TA I found the project east to understand. I believe when you understand the problem it become easy to solve it.

I found positioning the regex difficult because one wrong positioning changes the output drastically. Other than this first phase was easy or may I say ok.

I learn a lot about ply, more about regex, positioning of regex, more clear knowledge of tokens.