1. What is syntax analysis and how is it related to other parts in compilation?

   Syntax analysis is a phase of code the compilation processes run by code compilers. It comes after the lexing phase of the code compilation. While in the lexing of the code the tokens ("words") that make up the code are defined, in the syntax checking it's checked if the tokens form logical structures called non-terminals. In this phase the rules on how the tokens and non-terminals interrelate to each other are established.

2. How is the syntactic structure of the language expressed in the PLY tool? I.e., what parts are needed in the code and how are they related to syntactic rules of the language?

   The syntactic structure of the sheetscript language has been defined using the *Backus-Naur form*- metasyntax notation (BNF) in the PLY tool. For every non-terminal we wanted to define, a function was implemented that complied to the PLY tool. Every function that represents a non-terminal in our sheetscript language, starts with a "p_" and gets "p" as its argument. In the function The BNF- notation is written inside a multiline string.

3. Explain in English what the syntax of the following elements mean (i.e. how would you describe the syntax in textual form):
   1. Sheet variable definition

      A sheet is defined so that first a sheet token is identified at the start of it. After that there should be a sheet identifier token. After that there is the optional sheet initialization non-terminal.

   2. Function call

      A function call is a structural unit that is formed by a function identifier (FUNC_IDENT) token followed by "[" token followed by nothing or the arguments non-terminal followed by "]" token. The *arguments* non-terminal consists of zero or more arguments (arg_expr non-terminals) that are separated by comma (COMMA) tokens. The argument may be a scalar expression, range expression or a sheet identifier.

   3. Sheet variable definition with initialization list ("{…}").

      Its defined like the sheet variable but it has the optional sheet initialization. It comprises of an equals "=" sign followed by sheet initialization list or multiplication between two int literals. The sheet initialization list is comprised of curly brackets that have one or more sheet rows between them. A sheet row is comprised of simple expressions separated by commas. And so on.

4. Answer the following based on the syntax definition:
   1. Is it possible to define a "nested" function, i.e. to define a new function inside another function? Why?

No, nested functions are not allowed in sheetscript. This is due to the fact that the function definition is such that there is no possibility of a function existing inside another function. There is no recursion in the function definition.

2. Is it syntactically possible to perform arithmetic with integers (1+2)? Why ?

   It is because we have defined that a sheet can be initialized by performing arithmetic with two int literals. (BNF notation: sheet_init EQ INT_LITERAL MULT INT_LITERAL)

3. Is it syntactically possible to initialize a range variable with a decimal value (range _rng = 2.0+3.0)? Why?

   No it is not because in the BNF “range_definition ::= RANGE RANGE_IDENT [EQ range_expr]” the only possibility to initialize a range is to give a range expression. And the range_expr consists of options that do not include anything even resembling “range_expr : DECIMAL_LITERAL PLUS DECIMAL_LITERAL”.

4. Are the following allowed by the syntax: xx--yy and --xx? Why?

   --xx is not allowed by the syntax, because there is no rule defining that it is ok to have a double negation before an expression. One minus sign on the other hand is accepted. xx--yy can be accepted because simple expression (simple_expr) is defined so that it accepts a term and a plus or a minus sign and a term. Term can be negative so this way the yy--xx syntax is allowed.

5. Can comparisons appear in a sheet variable’s initialization list (sheet SS = { 1.0 < 2.0 })? Why?

   Not like in the example, because the sheet_row only takes in simple_expr non-terminals. They do not allow comparison. Comparison would require a scalar_expr. But on the other hand it seems to me that the initialization lists can have comparisons in them but they would be between parentheses that are between the curly brackets.

6. How is it ensured that addition/subtraction are done after multiplication/division?

   Addition and substraction are done on an upper level than multiplication and division. Mult/div is done in the term non-terminal by multiplying or dividing two factors but add/sub is done in the simple_expr where terms are added or substracted.

7. In SheetScript, statements and definitions are not separated by semicolons (like in Java/C++) or line breaks (like in Python). How does the syntax known when one thing ends and another begins?

   Tokens are separated by whitespaces. With the only exception being comments which start and end with the “…” marking. The tokens are recognized with regex and from the tokens the non-terminals are constructed. Whether an expression begins or ends is defined on the non-terminal level. When we discover that a continuum of tokens and non-terminals forms a syntactically correct structure we continue building on that until there is nothing to add (doesn’t fit in any higher level non-terminal). That’s how it’s figured if it’s the beginning of another structure.

5. Please mention in the document if you didn’t implement functions (i.e. you are ok with passing with the minimum grade).

   I implemented all functions (well I hope).

6. What did you think of this assignment? What was difficult? What was easy? Did you learn anything useful?

   The assignment was fun. It was interesting to learn more about the functioning of a compiler. The BNF- notation was surprisingly easy to grasp but the PLY yacc took more time to sink in. Of course, I didn’t even yet fully understand how it works but I internalized enough to work through the assignment. I think that it might be useful in the future that I somewhat know how a compiler operates and I’m also learning some python on the side which is great.