Modules used

The modules used within my program are re, graphviz, math, random, and sys. The re module is used in order to match and search for regular expression. This is used to error handle the input file. The graphviz module is used to create the graph itself with nodes connected by edges. This module also allows the graph to be rendered onto an image file. The math module is used to round up for a certain calculation in the program while the random module is used to produce a random id for certain nodes. To exit the program early the sys module is imported and the sys.exit() function used.

Instructions for graphviz

In order to use graphviz and have it correctly imported into my program; the user must follow a set of instructions so that the program can run as intended.

1. Go to the graphviz website and download the msi installer for windows.

2. Once the msi installer for graphviz has been run, add the bin folder path for graphviz to USER PATH which is found by opening file explorer, right clicking this pc, properties, advanced system settings and then going to environment variables.

3. Once this has been done you should install graphviz through pip just in case. There are a few ways to pip install, with some of them being 'pip install graphviz', 'python -m pip install graphviz' or 'py -m pip install graphviz'.

4. If problems persist then add the path of your pip installation to your PATH system variable.

Things to avoid

You should avoid making terminals with the corresponding values:

FORM

PRED

TERM

NEG

CONN

QUANT

CONS

VAR

This is because they are non-terminals within the grammar that I have used and therefore problems may arise when using these are terminal values.

In addition to this, for predicates the arity must be a single digit between 1 and 9 otherwise my code will not work correctly as I had spliced with certain indexes.

How the program runs

The way to run the program is by opening command prompt, cding to the program directory and typing in 'python dpnx39.py'. You will then input a file name in which the program will do the rest.

Grammar file

The grammar text file is an output file produced by the program when looking at the input file. The production rules are concatenated but they are displayed line by line as opposed to side by side as it is much easier to read. The file contains the set of rules used by the parser in order to determine if the formula given is valid or not. At the bottom of this file contains the set of terminals and non-terminals.

Parse tree

The parse tree that has been created seems to be quite accurate, this being based of on me solving by hand the example files that you had provided for the assignment. The parse tree has been made while the parser is run. So, while the parser is checking for formula validity, the parse tree is being slowly built along the way. The parser that has been used is recursive descent in which the parser goes through grammar rules that had been produced earlier into the program and recursing onto the FORM non-terminal function.

Log file

The log file will only contain one error message as the program stops and records the first error found. This is only the case when looking through input file to check if it is valid. This is because once the first error is found it assumes everything else would not work out when parsing. However, during the parsing stage multiple errors can be logged into the log file as it is telling the user the problems with the formula itself and not the input file.

For empty sets the program assumes that the input file at least contains the set name. So, for empty variables, the input file must contain 'variables:'. This is because the program will output an error message if it does not find 7 sets within the text file.

Other error messages include duplicate terms in different sets, if number of parentheses are not equal, if the predicate form or any other form for the sets is invalid, etc. When tokenising, the log file would contain a more detailed error message telling the user what symbol is causing a problem and where in the formula you can find it.

The general errors that you will find in the log file while the parser is parsing through the formula once the input file has been deemed valid are errors such as the program expected this symbol. For example, "Expected VARIABLE" would be one.