School of Computing Electronic
Engineering and Mechanical & Manufacturing Engineering

Compiler Construction
Continuous Assessment

Name	Course	Student ID
Cormac Duggan	CASE	17100348

For this project, apart from antlr and Java, there were 2 necessary files that needed to be created in order to parse the cal language successfully. These two files were the cal.g4 file and the cal.java file.

The cal.g4 file which was the grammar rules for the language was the first thing that I worked on. Using the rules defined in the pdf provided for the cal language I rewrote them in a format which would be legible to the antlr interpreter. Some of the rules provided were difficult to decipher as the pdf didn't make it very easy to determine which terminals were bold or not. For example condition rule 4 was a particularly rough one as the difference between bold and non bold pipes and parentheses appeared to be maybe a pixel difference. After finishing the rules I added the fragments for every letter as it was more straightforward than picking out which letters we need and only using them. Following that I added definitions for each terminal in the grammar including numbers, identifiers, two types of comments and white spaces. The last few of which would prove to be a little finicky. After completing the grammar it was compiled using antlr4 and ran through some test cal files to ensure there weren't any errors in the language. One issue I found with the grammar was that any occurrence of a + b + c would not work properly, however the definition of the language did not include this situation as a hypothetical. There can be an occurrence of (a + b) + c or something to that extent where parentheses are included appropriately, but without them in a statement would give an error. Additionally in the grammar there was an issue with the expression and fragment rules that caused left recursion that needed to be addressed. The solution for which was simply adding parentheses around the recursive expression call at the end of the fragment rule. This meant that only expressions contained inside parentheses could recurse back to the expression rule giving a finite amount of recursions.

The second file to be made was the parser file which was titled cal.java. In the java file similar to the examples provided in the course a pair of if statements for reading files is used to take in a file if one is provided in the arguments. After that the lexer and parser are initialized just the same as before in the examples. After this however, is where the file differs slightly from the examples. As the assignment says there should only be 2 potential outputs the default error listener attached to the parser is removed as it will give back error information if they exist in the cal file. After removing the listener a new one is added using the BaseErrorListener class in antlr. This class allows error listeners with custom output if you override the empty functions. So I did just that and override the syntax error function and replaced it with the appropriate output for a file that did not fully parse. After an error occurs the parse quits and if no error occurs the output will say the file has successfully been parsed.