Assignment 1
A Lexical and Syntax Analyser for the CCAL Language

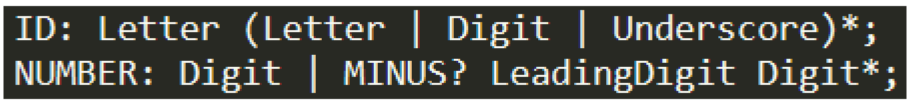
Name: Jason Boylan
Student Number: 18342986

ccal.g4

Tokens and Fragments

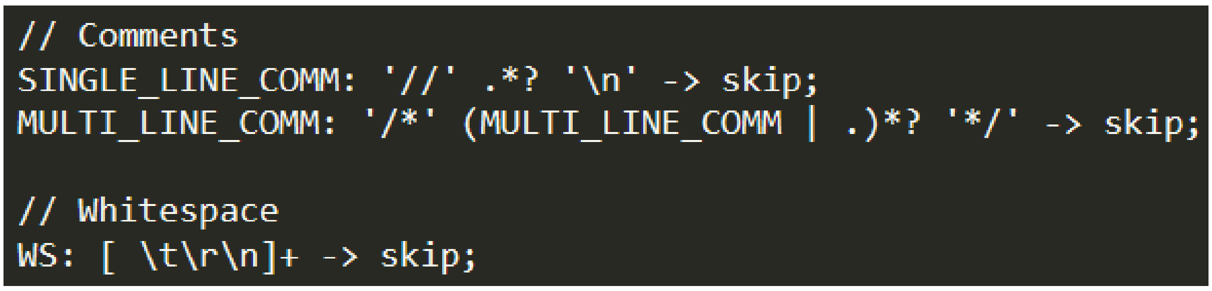
I began the assignment by creating the grammar for the CCAL language. I based some of this off the notes and the tutorial videos. I used the parser generator ANTLR4 to create the grammar.

The first thing I did was to name the grammar as CCAL and add case insensitivity. After this, I looked at the *ccal.pdf* document and included the reserved words and created the various tokens that were required. I created fragments for letters, digits and underscores in order to create an ID (Identifier) token. I also included a NUMBER token for integers that excludes leading zeroes.



```
ID: Letter (Letter | Digit | Underscore)*;
NUMBER: Digit | MINUS? LeadingDigit Digit*;
```

I then made sure that comments (both single- and multi-line) and whitespaces were skipped.

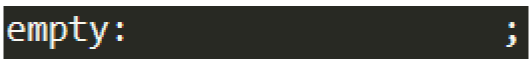


```
// Comments
SINGLE_LINE_COMM: '//' .*? '\n' -> skip;
MULTI_LINE_COMM: '/*' (MULTI_LINE_COMM | .)*? '*/' -> skip;

// Whitespace
WS: [ \t\r\n]+ -> skip;
```

Non-Terminals

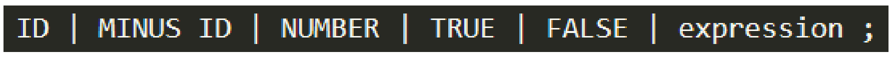
Upon defining the tokens and fragments to create a baseline for the grammar, I began to add non-terminals. There was a list of these in the PDF that I made sure to include. I also had to add an *empty* statement, to account for any statements that contained nothing.



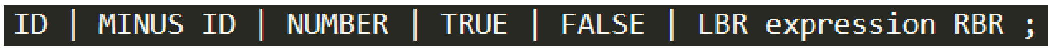
```
empty:               ;
```

Finally, I was getting the following error:
`The following sets of rules are mutually left-recursive [expression, frag]`

I corrected this by making this change to the `<frag>` rule:



```
ID | MINUS ID | NUMBER | TRUE | FALSE | expression ;
```

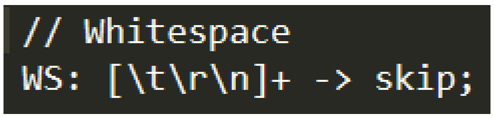


```
ID | MINUS ID | NUMBER | TRUE | FALSE | LBR expression RBR ;
```

ccal.java
Once the grammar was completed, I began working on the *ccal.java* file in order to create a lexer, parser and token stream. I created this by following the *draw.java* and *bp.java* examples from the Zoom sessions.

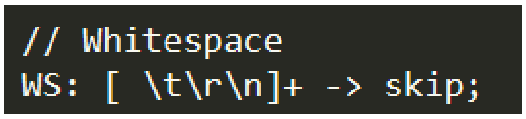
test.ccal
As I was working on the assignment, I began creating test files based on tests from *ccal.pdf*. My first test, *test-1.ccal*, tested case insensitivity, comments and functions. It failed when I first ran it.

I was receiving this error: `token recognition error at: ' '`
The issue was in my whitespace grammar as I'd forgotten to include a space. Once this was resolved, the test passed fine.



```
// Whitespace
WS: [\t\r\n]+ -> skip;
```

=>



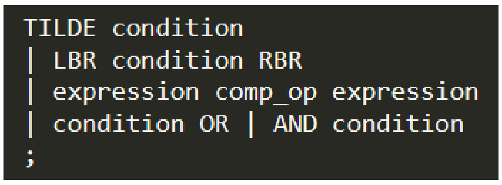
```
// Whitespace
WS: [ \t\r\n]+ -> skip;
```

The second test, *test-2.ccal*, passed first-time. This file demonstrated the different scopes.

The third test, *test-3.ccal*, demonstrated the use of a variety of functions. It failed initially with these errors:

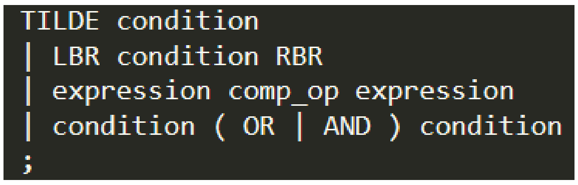
```
line 8:13 mismatched input '&&' expecting {'{', '||'}
line 9:4 mismatched input '{' expecting {'{', '||'}
line 16:8 mismatched input '{' expecting {'{', '||'}
line 22:22 mismatched input '&&' expecting {'{', '||'}
line 38:4 mismatched input '{' expecting {'{', '||'}
```

This was due to an issue with my `<condition>` rule. I made the following change and the test passed:



```
TILDE condition
| LBR condition RBR
| expression comp_op expression
| condition OR | AND condition
;
```

=>



```
TILDE condition
| LBR condition RBR
| expression comp_op expression
| condition ( OR | AND ) condition
;
```

Once these tests were running, I was satisfied with the grammar.