Assignment Submission

Module Title	Compiler Construction
Module Code	CA4003
Assignment Title	Semantic Analysis and Intermediate Representation for the CCAL Language
Submission Date	12th December 2016
Name	Tríona Barrow
Student Number	11319851
Module Co-Ordinator	David Sinclair

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

Semantic Analysis and Intermediate Representation for the CCAL Language

I started with the initial grammar I created for the first assignment, simply changing the file type from .jj to .jjt. From there I updated the user code to represent close to the ExprLang.jjt example, added more options, and updated the grammar in line with generating an abstract syntax tree.

- *Options*

I retained the `IGNORE_CASE` flag from part one, and added the following:
`MULTI = true;` - to allow a multi node parse tree to be made
`VISITOR = true;` - to allow visitor class to be used, needed for semantic analysis
`NODE_DEFAULT_VOID = true;` - to push nonterminals onto the stack by default

- *User Code*

I renamed the parser to CCALParser, then copied the layout from ExprLang for the file reading and printing the AST. The semantic visitor file, and intermediate representation would also be printing from here - however I was not able to complete these. The visitor class is called and started correctly - however the methods in the file all return null, so this will not print anything.

- *Token Definitions*

The general token definitions were using the same logic as in the initial .jj file, however it needed to be updated to interact and build an AST. I picked up on a mistake from the first assignment - in `var_decl()` and `const_decl()` I forgot to call any identifier, so I was able to catch this and fix it to match the language specification.
From there I was reorganising definitions,I created one for the `<ID>` token, and refactored `bin_op()` and `comp_op()` into the definitions that used them (both only called by one definition).
I also removed the attempts to catch epsilon, from speaking to classmates about the assignment they suggested `{}` instead to catch it. I used this in place of the previous `<EMPTY>` token in the token definitions.

- *Problems*

- CCALParser.jjt

In the test file given, the final line of code in the file is not parsed correctly - result = multiply(arg_1, arg_2); - this should be handled appropriately from the first definition in statement().

```
void statement() #Statement : {Token t;} {
      (identifier() (<EQUALS> expression() | (identifier() <LBR>
arg_list() <RBR>)) <SEMIC>)
    | (<LFTBRC> statement_block() <RGTBRC>)
    | (t = <IF> condition() <LFTBRC> statement_block() <RGTBRC>
<ELSE> <LFTBRC> statement() <RGTBRC>)
    | (t = <WHILE> condition() <LFTBRC> statement_block()
<RGTBRC>)
    | (t = <TKNSKIP> <SEMIC>)
}
```

However, when trying to change it to match that line, it began having difficulties parsing thins further up the test file - causing both choice conflicts and parse exceptions to be generated. I was unable to find a fix for this, however with this line removed from the test file, the AST is printed with no issues.

- Symbol Table

An attempt was made towards this, however I was unable to resolve the errors on time for submission, so sections related to this are commented out.

- SemanticCheckVisitor.java

There is the barebones for a working visitor in there, however it doesn't return anything with the nodes it visits. It just does a node accept for any child nodes and returns null.

- Three Code Representation

As I was unable to finish the earlier parts, there is no attempt towards this section.