Juan Carlos Hernandez Puebla
Professor Steinmetz
CSC 450
December 18, 2015

Test Document

This document describes the tests implemented for the Scanner and Parser.

Scanner

The Scanner was tested though various txt files, which tested for correct identity of valid and invalid tokens, as well as its token type and lexeme. This was done by iterating through the tokens found in the test file and asserting the values for token, token type, and lexeme. The first test file named input.txt tested for general recognition of different Token Types of the Pascal language. The Token Types included were ID, EQUALS, LESS_THAN, and NUM. The Scanner correctly asserted the expected values for token, token type, and lexeme. Included in this test file was the # character, which is a symbol not part of the Pascal language. Hence, it asserted that the token be false, the token type be null, and the lexeme be "#". The test that followed tested for more specific lexical conventions. For example, the second test named badComment.txt, tested for a bad written Pascal comment. The test file included a second { symbol within the initial { symbol and the ending } symbol. This represents an error and so the token was asserted to false, the token type was asserted to null, and the lexeme was "{". The next test was for a well written Pascal comment, where the entire comment should be ignored. Since nothing was written after the comment, the Scanner has reached the end of the file. Therefore, the token was asserted to false, the token type to null, and the lexeme to null. These assertions are made every time the Scanner reaches the end of a test file.

Parser

The Parser was tested by taking in a Pascal written program file as its argument. Then the Parser invoked its program function. If the program was recognized as a valid Pascal program, then a successful parse message was printed. The first test file named simplest.pas tested for the simplest valid Pascal program possible. It contained no declarations and no sub program declarations, just the following.

```
program f;

begin

end
.
```

This file was successfully identified as a valid written Pascal program. The next test file named singleDeclaration.pas, added a single declaration to the simplest valid program. The Parser successfully identified this file it as a valid program. The next test named declarations.pas, added more declarations of both integer and real types. The Parser identified it as a valid program as well. Eventually the Parser was tested with Pascal programs that included declarations, subprogram declarations, and compound statement, all together.

After the valid written programs were tested, then invalid Pascal programs were used to test the Parser. The first invalid test file named badDeclarations.pas tested for an invalid declaration due to its standard type being declared as a double. The Parser was able to identify the non-matching Token Type

for the standard type function. It also successfully printed out the line of where this error occurred. The rest of the test files tested for incorrect implementations of subprogram declarations and incorrect implementation of a compound statement.