Part 1: Written Report

Corbin T. Rochelle

October 3, 2022

1 Two Regular Expressions

The two regular expressions that I am going to discuss are the following:

```
\:          { return TOK_COLON; }
```

and

```
\{[A-Za-z 0-9\.\,\$\^%!&]*\}
```

In looking at (1), we see that this lexeme strictly pulls a colon out of the input stream and outputs the TOK_COLON token. The only real important note about this code is the backslash before the colon. This is needed because the colon is an overloaded character in the TIPS language.

In looking at (2), we see that this lexeme is a lot more advanced. While this syntax may look confusing, it is quite simple. This code simply looks for and pulls everything in the input stream that has any character in between an open and closed curly brace, our comment characters. There is no C code following the identification, meaning it is simply excluded from input stream.

2 The Development and Testing Process

The development process for this part of the lab was fairly easy as it only took me around two hours to code and test to completion. The first step was to review my code for the previous lab to remember the syntax of the rules. I then coded all of the singular character Lexemes, then attempted the multi character ones (from the top to the bottom of the provided pdf.) I began testing the code on the input1 and input2 documents after I attempted all of the rules. My testing process involved going through the test documents one-by-one and modifying my rules if the diff yields results. After the first rune through, I redid all of the tests to see if I messed up a previous test by modifying for one of the later tests.

3 Sources

The only source that I consulted during the development of my code was University of Oregon-Ohkosh's documentation. The purpose of looking at this page was to see the list of operators in the LEX language, so I knew what characters needed to be escaped.