Jesse Huss
001209444
February 3, 2020
CPSC 3740 Assignment 2

Question 1

In question one we are instructed to read through the write up on utilizing Flex and Bison to take an equation input from a user and display the results. The way this is implemented in the write up is quite simple compared to trying to create a custom parser and logic to handle all the operations. In this short report I will be summarizing the basic process of how this functionality was achieved using Flex and Bison as well as touch on the basic specifications necessary.

There are two main components to this exercise that contribute to the end result. Those two components are: the parser, and the scanner. Together these two components generate a program that is able to take user input of a basic mathematical equation, analyze it and return a result.

The first step is creating the specification file using Bison. This file has four main sections. The first is the headers, we add our inclusions and definitions here, it is important to note that they are not processed by Bison, but instead go straight into the generated file. The next section contains the tokens. This section is where we set up our operator precedence. Then we have the section containing the rules. This section lists the allowable forms of expressions, as well it contains some code to show what the expressions mean. Finally we have the section for user defined functions. In our code we just have an error display and the main function which runs the parse command and will also show errors. Once we have this all set up and ready to go our next step is to process the specification file. We do this by running the command: **`bison –d calc.y`**. What this command will do is generates two different files that will be used later. Now we can move on to the Flex part.

The Flex part of this exercise is what actually reads the input and processes it for bison to use. In order for this to all work smoothly we need consistent token definitions between Flex and Bison. We need to create a definition file for flex which has also got four main sections. First is the headers, very similar to the first section of the Bison specification file. Next we have a section for regular expressions. This section uses regular expressions to describe the types of things in the file, for example: what “white space” is. Next we have a section for rules, whose main purpose is to describe what the output for each type of thing encountered should be. In this section we want to ignore whitespace and comments, only sending the tokens to the parser. The last section just like in the Bison specification file is for user defined functions which for this exercise we have none. One this file is all written, we use the command **`flex –o calc.lex.c calc.lex`** to generate our function. The next step is just to compile everything, and watch the magic.

This exercise is really cool because it highlights the functionality and simplicity of using Flex and Bison to create programs for you by just defining rules.

Jesse Huss
001209444
February 3, 2020
CPSC 3740 Assignment 2

REFERENCES

http://www-h.eng.cam.ac.uk/help/tpl/languages/flexbison/