The final programming assignment, Mini JSON Parser, is C++ program that reads in the string as the input from the memory, parses the string into respective data types and stores them, and again prints out the final result in string format with the help of ToString() function.

The Mini JSON parser has IJSON\_document.hpp, JSON\_documents.hpp, JSON\_string\_extension.hpp and our self implemented Datatypes.hpp as header files. With the help of the extension library and the other header files written by Professor Cordova, we are able to implement our header file and run the program.

The Datatypes.hpp header file contains a Datatypes, as a parent class with a pure virtual ToString() function and a destructor. It’s child classes contains various data types that we need to parse, such as:

* Integer
* Double
* Boolean
* String
* Array

Each of the child classes has a constructor that reads in the string, ToString() function to return the output and a destructor to prevents memory leaks. The constructor reads in the string from the memory and parsed into its respective data type. The parsing is called from the Insert() function in the JSON document that compares and differentiate which string goes where. The ToString() function, converts the parsed data into a string again to display as output.

The JSON document header files contain canonical member functions like default constructor, copy constructor, move constructors, assignment operators and a destructor. The other important functions that we implemented are Insert(), Remove(), ToString() and Traverse(). The Insert() function reads in the <key, value>(<K, V>) pair from the input string and is implemented such that it goes through each condition that it would separate the string.value() and store it in the heap that will parse it into its respective data type like bool or array or ints, etc.

The Remove() function, will read in the string.key() and search the string of <K, V> pair for the first element that is the same as the key to be removed. This function erases both the first and second elements from the data.

The String() function here calls all the pure virtual ToString() from Datatypes class that gets all the parsed data into a string and the output code will look very similar to the one that we read. Traverse() loops through the input string.data() and searches <K, V> pair that the user requires. And the destructors loops the data delete the second element from all the given input data.