Bartolomeo Gisone: ------------

Victor Chung: ----------

Intro:

The main objective of the lab is to program a basic finite state machine in C. The program must read lines from a .txt file and parse through those lines, getting the nth word each individual line. The program gets the n value, through user input in the command line.

Explanation:

Our code works as follows:

Firstly, the program is initialized to the FILE\_READING state, where it opens a txt file. If the file does not exist, the program returns an error and closes. If the file exists, the program prompts the user for a number n, where n is the position of the words, which are to be extracted by the program. If the given n is less than 0, returns an error and closes file.

If valid, the state changes to WORD\_PROCESSING and the program runs a while loop for every line in the file. While going through line by line, the getNthWord function, counts the number of spaces and returns the Nth word in a line. If the nth word does not exist, returns nothing, else, returns the nth word. If the word is longer than 0 (the word exists), then we remove \n and \EOF values and print the word.

The State changes back to FILE\_READING, and the program cycles through the words, until it reaches the end and closes.

The program uses states to ensure to solve the problem of reading through lines and correctly gives the word without any unnecessary \n or \EOF values.

Conclusion:

When completing the lab, the focus on states made us think about how we could solve the problem in a way in which a finite state machine was used. The turnstile examples in the lab document helped us figure out how to apply finite state machines in a simple fashion. We came into trouble when creating the getNthWord method, as it was difficult for us to find the nth word in a generic sentence, where we don’t know how long the sentence might be, or if it’s an empty str. Immutable objects like strings and pointers make it a more difficult to find certain words.