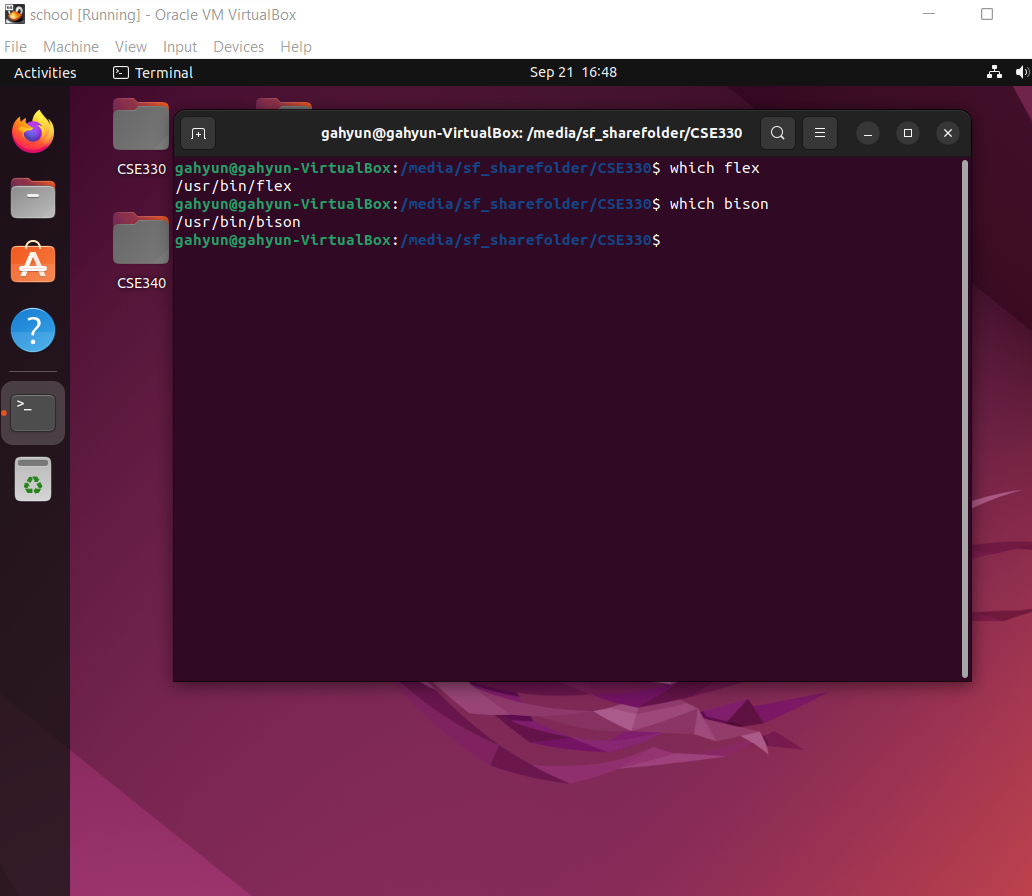
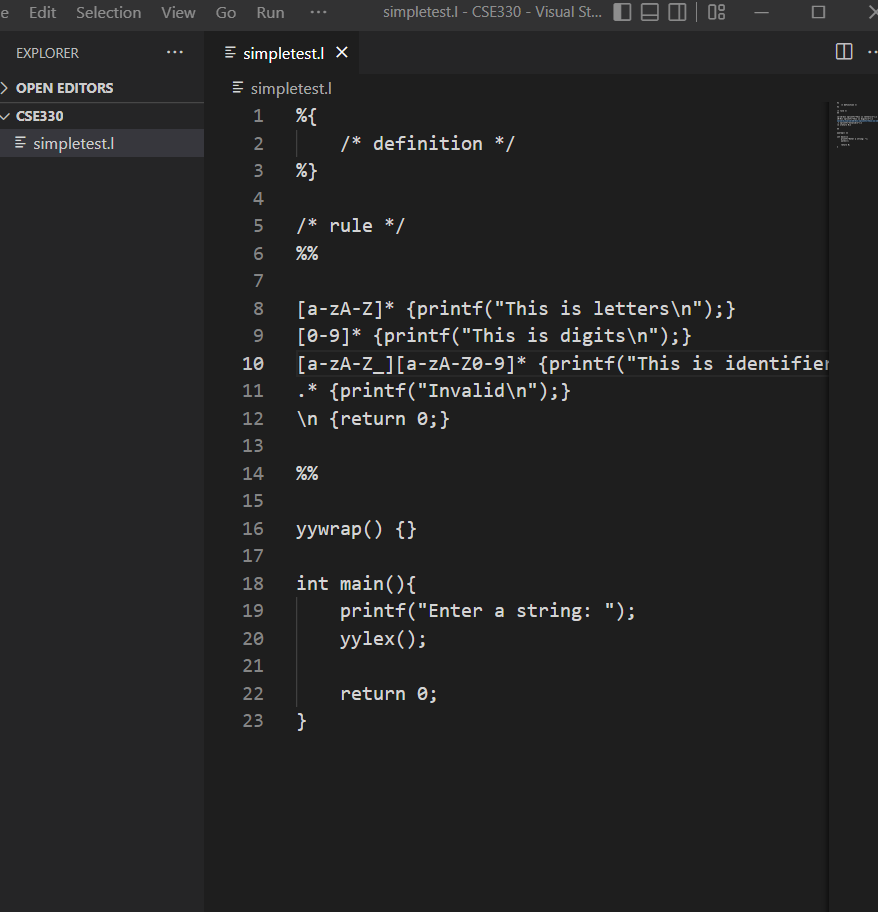
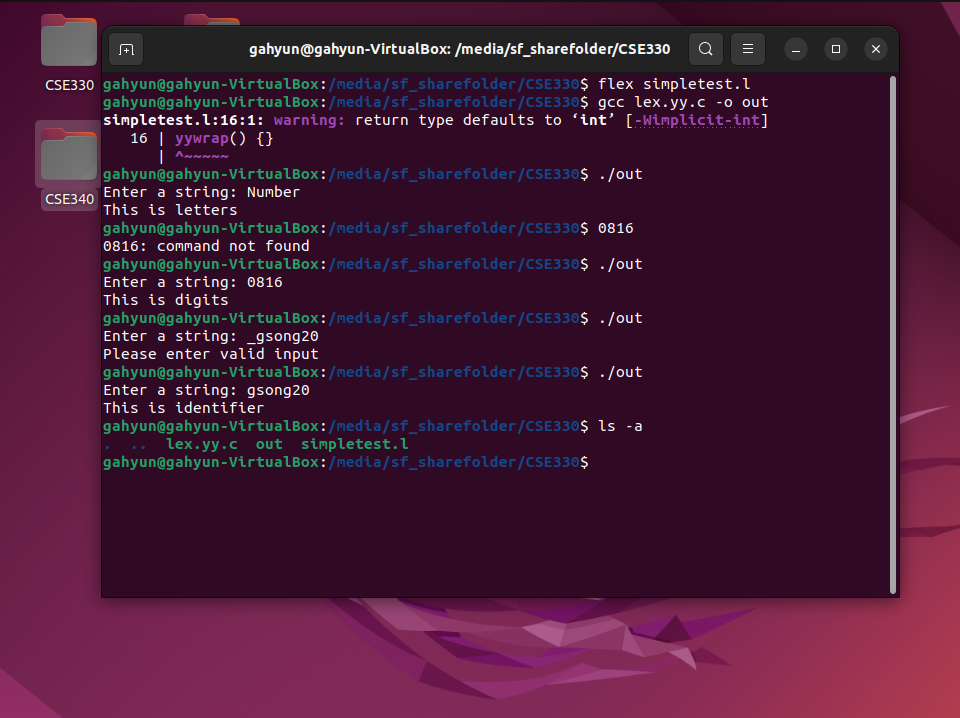
1. **Describe in a short write up (150-200 words) your process of setting up and installing Flex and Ubuntu/Unix and what IDE you will be using.**

* I worked this assignment on Virtual Box with Ubuntu. As you can see below, I installed flex and bison on Ubuntu. I installed them by writing “sudo apt-get install flex bison” on Ubuntu.
* I also installed Visual Studio Code on my host window as the IDE, and I created the .l file on it, and shared the file with Ubuntu on VM. To share the file from host window to VM, there were a few steps that I had to take. I installed Guest Additions on VM, then I added the directory of the folder I want to share with. Also I needed to add my ubuntu account to the vboxsf group to access the sharefolder. After all of those steps, I can use my host window and VM together.



* When I created the “.l” file on VScode, I analyzed input with flex and compiled with gcc on Ubuntu on VM, which generated lex.yy.c file and output file from .l file.



1. **Write a short description of how the Lexer works in compiler design and the files that were generated when you ran you lexer. Describe the tokens you will need to build a full language (this will be project 2). Make sure to describe theoretical principles, design practices, and implementation strategies of #compiler design that you will put into practice.**

The lexer is a program that perform Lexical Analysis in compiler design. The lexer takes the modified source code which is in the form of sentences. In this project, I created tokens for "char", "write", "return", ";", ",", ", "=", and {NUMBER}. For instance, if you put the word "return" each respectively, r,e,t,u,r,and n are meaningless. However, "return" becomes meaningful through tokenization. When this token steps through the lexer, it analyzes the meaning and returns the "return" word. This token is analyzed like {type: , value: "return", child:[]} and transfers to the Parser. As I explain, other tokens that I made also step through lexer to analyze and return it.

For the new tokens I will need to add different keywords to the language to be able to identify what type of which function they will perform in the language that we are creating. I will be doing this by using the same method that I did for making the return token.