Project 3 Report

Name: Kyle Fiori

Class: CMSC-430

Assignment: Project 4

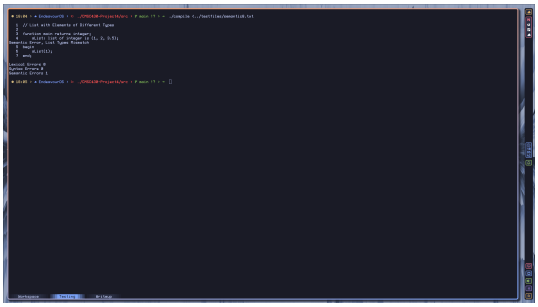
Date 2024-08-06

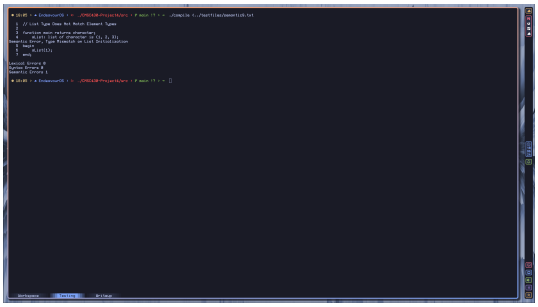
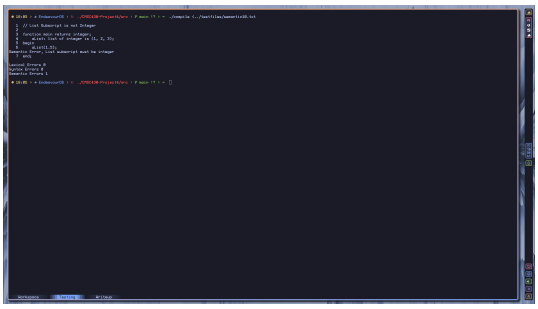
Approach

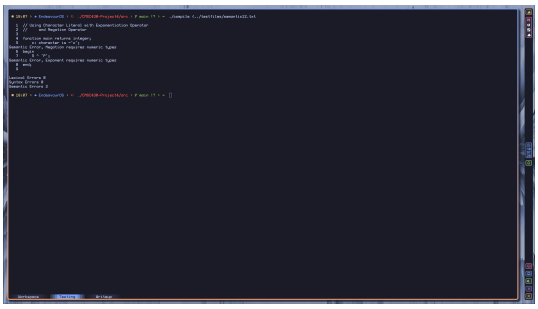
I followed the recommended approach to a 'T' with this project. I started with the skeleton and added all additional tokens and semantic rules from Project2. Once completing these, I tested the "valid" tests that were included with the materials. There were some warnings at compilation but they had no affect on the output so I ignored them for the time being. I began to tackle the new rules 1 by 1. At first I felt a little "in the dark" about how the given functions worked in types.cc. After trial and error I began to get the rules 1 by 1. The biggest issue was figuring out when a function needed to return a type and be void. The next holdup was figuring out how to catch duplicate variables all the way at the end. This simply involved adding a new method at the end of parser.y to pass the correct information to a function in types.cc. Once implemented I cleared up some of the extra warnings with the compile.

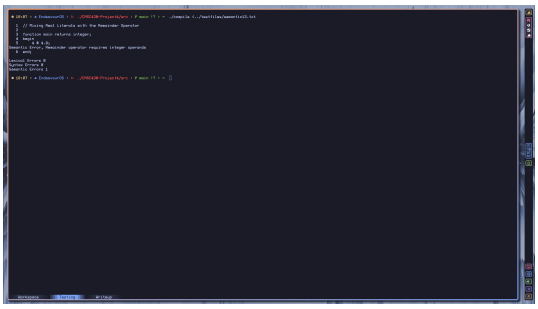
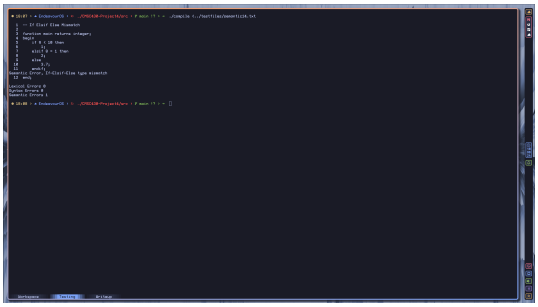
Testing

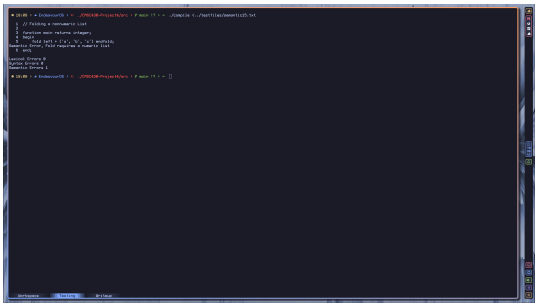
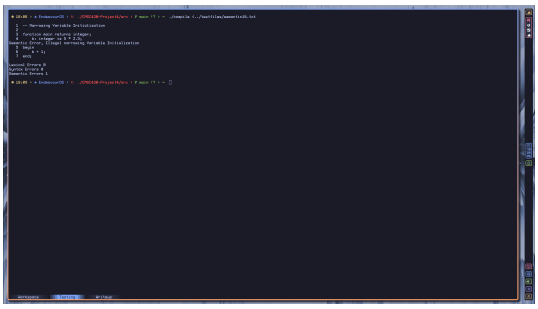
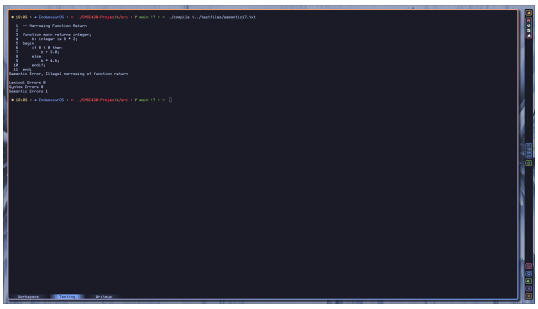
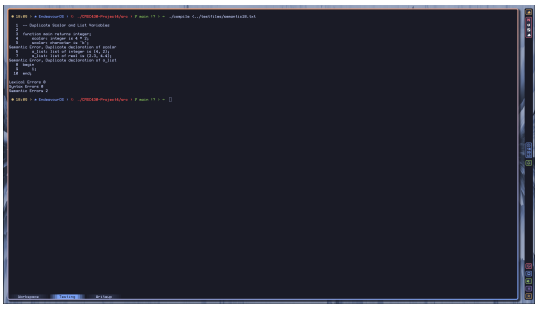
I ran every test case given with the project that added additional features. The test results are as seen below:

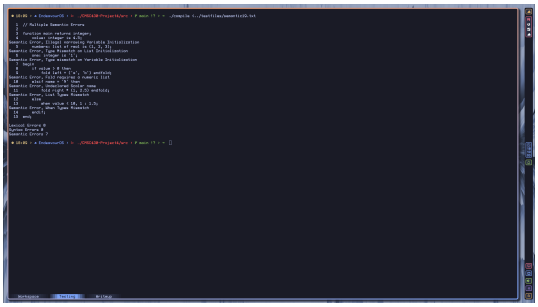
Test#	Description	Screenshot	Pass/ Fail
1	• Single semantic error expected • Features tested: ◦ List with mismatching element types		Pass

2	• Single semantic error expected • Features tested: ◦ List initialization with different element types		Pass
3	• single semantic error expected • Features tested: ◦ List reference with non integer subscript		Pass
4	• Single semantic error expected • Features tested: ◦ Relational operator where character is compared to numeric type		Pass

5	• Two semantic errors expected
	• Features tested:
	  ◦ Arithmetic negation used on a character
	  ◦ Character used with exponentiation operator
			Pass

6	• Single semantic error expected • Features tested: ◦ modulo operator used with real type		Pass
7	• Single semantic error expected • Features tested: ◦ If statement return types mismatching		Pass

8	• Successful compilation expected • Features tested: ◦ Non numeric list used on fold statement		Pass
9	• Single semantic error expected • Features tested: ◦ Narrowing on variable initialization		Pass
10	• Single semantic error expected • Features tested: ◦ Narrowing on function return		Pass
11	• Two semantic errors expected • Features tested: ◦ Duplicate initialization on variables		Pass

12	• Seven semantic errors expected
	• Features tested:
	  ◦ Narrowing on variable initialization
	  ◦ List initialization type doesn't match list
	  ◦ type mismatch on variable initialization
	  ◦ Fold with non numeric list
	  ◦ referencing undeclared scalar
	  ◦ list with differing element types
	  ◦ type mismatch on when statement return
			Pass

Lessons Learned

This project gave me a better understanding of error checking and reporting within a compiler (or at least how to implement it with bison). As previous projects I think the most valuable skill gained was the ability to analyze a given code base, determine how it process information, and implementing new features to it. Overall I think this class did an okay job of introducing compilers. Enough hold a conversation and use the correct jargon about them. If I was not using Flex and Bison I think I would be pretty lost on implementing one myself. I'm not sure if these tools are industry standard or not for creating new compilers.