**Project Approach**

While trying to understand how to write and what to write for our week 5 discussion post, I ended up going ahead and starting on project three. As a lot of the course content for week five and week six went hand in hand. To be able to understand the some of these concepts, I thought working with project three for a hands-on approach would allow me to get a better understanding of what week five’s discussion post was asking. It also allowed me to get a head start of project three.

Reviewing the content for week six gave me a better grasp of how to construct the interpreter, and how the semantic rules are structured in the bison file (parser.y). I think the videos in the content for week six is where I got a lot of help for project three, which ultimately helped me complete the discussion post for week five. While following along with the course content and applying the concepts to project three, made me really want to finish project three before writing my discussion post (at least finish enough to compile my program and receive a calculated result). Once I got project three able to read and compiled a program that met discussion five’s instructions, I started further work on project three.

I ran into a few issues (will discuss further in the lessons learned section) that required a little more time and research from outside sources. However, for the most part I was able to get most of the information from our course content, mainly in week five and six.

**Test Cases**

Test Case 1:

* Integer, Boolean, Real
* Logical ops: and not
* Arithmetic ops: + \* / rem \*\*
* Relational ops: > < >= <= /= =
* If/Else statements (nested)
* Case statements
* Reduce statements
* Multiple command line inputs/parameters
* Multiple variables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Num.** | **x** | **y** | **z** | **Result** |
| Test 1 | 0 | 3 | false | 7 |
| Test 2 | 1 | 2 | false | 25 |
| Test 3 | 2 | 2 | false | 0 |
| Test 4 | 3 | 1 | false | 3 |
| Test 5 | 4 | 4 | false | 18 |
| Test 6 | 5 | 11 | true | 30 |
| Test 7 | 5 | 11 | false | 23 |
| Test 8 | 6 | 0 | true | 22.5 |

Screen Shots for Test Case 1:

* Test 1

|  |  |
| --- | --- |
|  | (passing parameters x=0, y=3, z=false) This test should enter the first two if conditions and then move into the case statement. Where x=0, and returns the ‘when 0’ case (Line 22). We have:  even\_num + odd\_num  (0 \* (0 + 1)) + ((2 \* 3) + 1) = 7 |

* Test 2

|  |  |  |
| --- | --- | --- |
|  | (passing parameters x=1, y=2, z=false) This test should enter the first two if conditions and then move into the case statement. Where x=1, and returns the ‘when 1’ case (Line 23). We have:  odd\_num \*\* even\_num  ((2 \* 2) + 1) \*\* ((1 + (1 \* 1) = 25 |  |

* Test 3

|  |  |  |
| --- | --- | --- |
|  | (passing parameters x=2, y=2, z=false) This test should enter the first two if conditions and then move into the case statement. Where x=2, and returns the ‘when 2’ case (Line 24). We have:  even\_num rem x  (1 + (1 \* 1)) rem (2) = 0 |  |

* Test 4

|  |  |
| --- | --- |
|  | (passing parameters x=3, y=1, z=false) This test should enter the first two if conditions and then move into the case statement. Where x=3, and returns the ‘others’ case (Line 25). We have:  odd\_num / y  ( (2 \* 1) + 1) / (1) = 3 |

* Test 5

|  |  |
| --- | --- |
|  | (passing parameters x=4, y=4, z=false) This test should enter the first if condition, then move to the next if statement where it will hit the else condition. From there, it should go into the next if condition (Line 28), where it will then calculate the reduce statement (Line 29).  Reduce statement == 3 \* 6 = 18  (4) +2 = 6  3 |

* Test 6

|  |  |
| --- | --- |
|  | (passing parameters x=5, y=11, z=true) This test should enter the first if condition, then move to the next if statement where it will hit the else condition. From there, it will move to the next else condition, then into the next if condition (Line 37). Where z is true, and will return even\_num.  (5 \* (5 + 1)) = 30 |

* Test 7

|  |  |
| --- | --- |
|  | (passing parameters x=5, y=11, z=false) This test should enter the first if condition, then move to the next if statement where it will hit the else condition. From there, it will move to the next else condition, then into the following else condition (Line 38). Where z is false, and will return odd\_num.  ((2 \* 11) + 1) = 23 |

* Test 8

|  |  |
| --- | --- |
|  | (passing parameters x=6, y=0, z=true) This test should enter the first else condition (Line 43). As, y is zero and it will return real\_num.  Odd\_num \* 22.5  ((2 \* 0) + 1) \* 22.5 = 22.5 |

Test Case 2:

* Integer, Boolean
* Logical ops: and
* Relational ops: > <
* If/Else statements
* Case statements
* Command line inputs/parameters
* Multiple variables

|  |  |  |
| --- | --- | --- |
| **Test Num.** | **a** | **Result** |
| Test 1 | 0 | 0 |
| Test 2 | 1 | 0 |
| Test 3 | 2 | 1 |
| Test 4 | 3 | 0 |
| Test 5 | 4 | 0 |
| Test 6 | 5 | 0 |
| Test 7 | 6 | 0 |

Test Case 2 Screen Shots:

* + Test 1

|  |  |
| --- | --- |
|  | (parameters a=0) ) This test should enter the if condition, then enter the case condition. Where a meets ‘when 0’ (Line 17) will return f (f=false). |

* + Test 2

|  |  |
| --- | --- |
|  | (parameters a=1) This test should enter the if condition, then enter the case condition. Where a meets ‘when 1’ (Line 18) will return f (f=false). |

* + Test 3

|  |  |
| --- | --- |
|  | (parameters a=2) This test should enter the if condition, then enter the case condition. Where a meets ‘when 2’ (Line 19) will return t (t=true). |

* + Test 4

|  |  |
| --- | --- |
|  | (parameters a=3) This test should enter the if condition, then enter the case condition. Where a meets ‘when 3’ (Line 20) will return f (f=false). |

* + Test 5

|  |  |
| --- | --- |
|  | (parameters a=4) This test should enter the if condition, then enter the case condition. Where a meets ‘when 4’ (Line 21) will return f (f=false). |

* + Test 6

|  |  |
| --- | --- |
|  | (parameters a=5) This test should enter the if condition, then enter the case condition. Where a meets ‘others’ (Line 22) will return f (f=false). |

* + Test 7 🡪 Take screen shot for this case

|  |  |
| --- | --- |
|  | (parameters a=6) This test should enter the else condition (Line 24) will return f (f=false). |

Test Case 3:

Testing for this test case will cover:

* Integer, Boolean, Real
* Logical ops: and not or
* Arithmetic ops: + - \* / rem \*\*
* Relational ops: > < >= <= /= =
* If/Else statements (nested)
* Case statements
* Reduce statements
* Multiple command line inputs/parameters
* Multiple variables

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Num.** | **a** | **b** | **c** | **t\_f** | **Result** |
| Test 1 | 5 | 10 | 18 | false | 9.3 |
| Test 2 | 20 | 0 | 0 | true | 7.5 |
| Test 3 | 0 | 3 | 7 | False | -27 |
| Test 4 | 5 | 0 | 50 | True | 5 |

Test Case 3 Screen Shots:

* + Test 1

|  |  |
| --- | --- |
|  | (parameters a=5, b=10, c=18, t\_f=false) This test should enter the first if condition (Line 19) and return:  ((a \* c) + (c rem a )) / b  ((5 \* 18) + (18 rem 5 )) /10  ((90) + (3)) / 10 = 9.3 |

* + Test 2

|  |  |
| --- | --- |
|  | (parameters a=20, b=0, c=0, t\_f=true) This test should enter the first else condition, then hit the next if condition (Line 22). Where it will return:  reduce \*  3  2.5  reduce +  1.0  = 1.0 + (2.5 \* 3.0) = 7.5 |

* + Test 3

|  |  |
| --- | --- |
|  | (parameters a=0, b=3, c=7, t\_f=false) This test should enter the first else condition, then into the next else condition. Where it will enter the case statement. While t\_f=false, it will hit ‘when 0’ (Line 32) and return:  (a - b) \*\* (b rem c)  (0 - 3) + (3 rem 7 )  (-3) \*\* (3) = -27 |

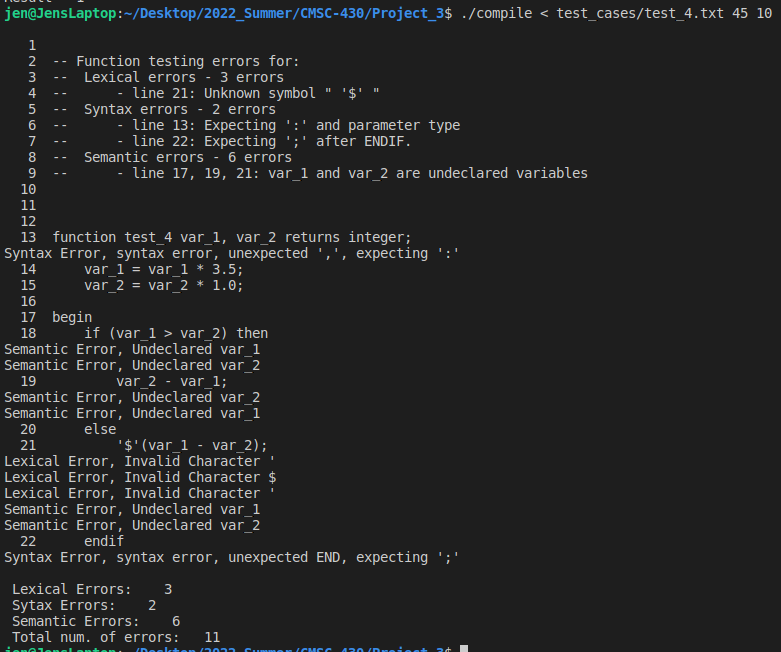
* + Test 4

|  |  |
| --- | --- |
|  | (parameters a=5, b=0, c=50, t\_f=true) This test should enter the first else condition, then into the next else condition. Where it will enter the case statement. While t\_f=true, it will hit ‘others’ (Line 33) and return:  5 |

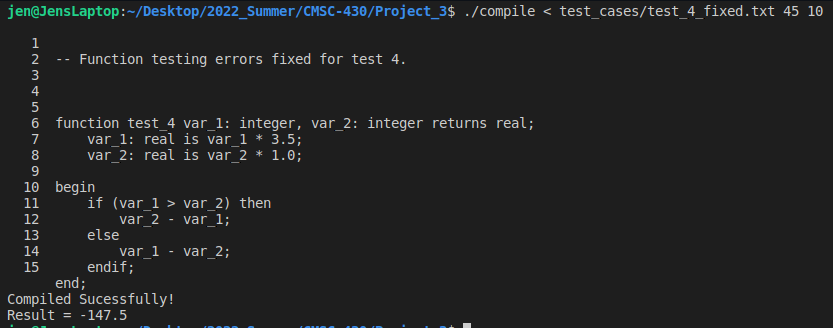
Test Case 4:

Testing for this test case will test errors for:

* Lexical errors - 3 errors
* line 21: Unknown symbol " '$' "
* Syntax errors - 2 errors
* line 13: Expecting ':' and parameter type
* line 22: Expecting ';' after ENDIF.
* Semantic errors - 6 errors
* line 17, 19, 21: var\_1 and var\_2 are undeclared variables



Test Case 4 (fixed errors)



**Lessons Learned**