Name: Riley Hickman
Date: 2/16/2014

Milestone Report3

Handwritten Answers to Milestone Questions:

(See Attachments) for grammar stuff, and the pseudo code for S() grew up and is now adult code.

Testing Assignment 1 –
Shortest length terminal word is [], 2. Next shortest is [1], 3.
Testing Assignment 2 –
Shortest length terminal word is [], 2. Thus n=1 is impossible with out error.
3: [1]
4: [[]]
5: [[1]]
Testing Assignment 3 –
Well in short go back words. Instead of eliminating types we would show all permutations of that length.

Specification (what do you think the purpose of this milestone is)

The purpose of this milestone is to convert the stream of tokens into a parse tree, and check its syntax validity against the languages grammar. Other objectives include; improving the symbol table; creating and testing the token, and tree data structures. The milestone also serves to provide Antonio and I experience with production system concepts, categorical style design, and practicing designing test files based on formal definitions.

Processing (how did you and/or your team go about solving the problem)

Well learning from last time, when we jumped on the code a little fast we spent more time in the design phase, this was an awesome idea by the way. We wrote down the entire grammar, and then removed all left recursion. Then we refined the grammar to make it easier to program. Once we were set on the theory we wrote out nearly all of the pseudo code for the parser. During some of this I spent developing and testing the data structures for the parser tree. Then we converted the pseudo code into real code. Wrote up all the test cases we had in mind then the function to easily use all of our test cases. Tested, refined, repeat.

Testing Requirement (how did you and/or your team test for correctness)

So testing ended up being quite hefty for this milestone. We found we needed roughly 48

Name: Riley Hickman
Date: 2/16/2014

cases at the time of writing this to get good enough coverage. Also we really wish we had a simpler way to set up automated test checking, since we are still checking the unit tests by hand. We had a couple of errors that were very challenging to fix since the expected fix's didn't work like in the case of, [5]5 which is invalid; the solution is really easy theoretically but when we pushed to code it was a hot mess (when the last] is found in T() check for any remaining charcters).

Retrospective (what did you learn in this milestone)

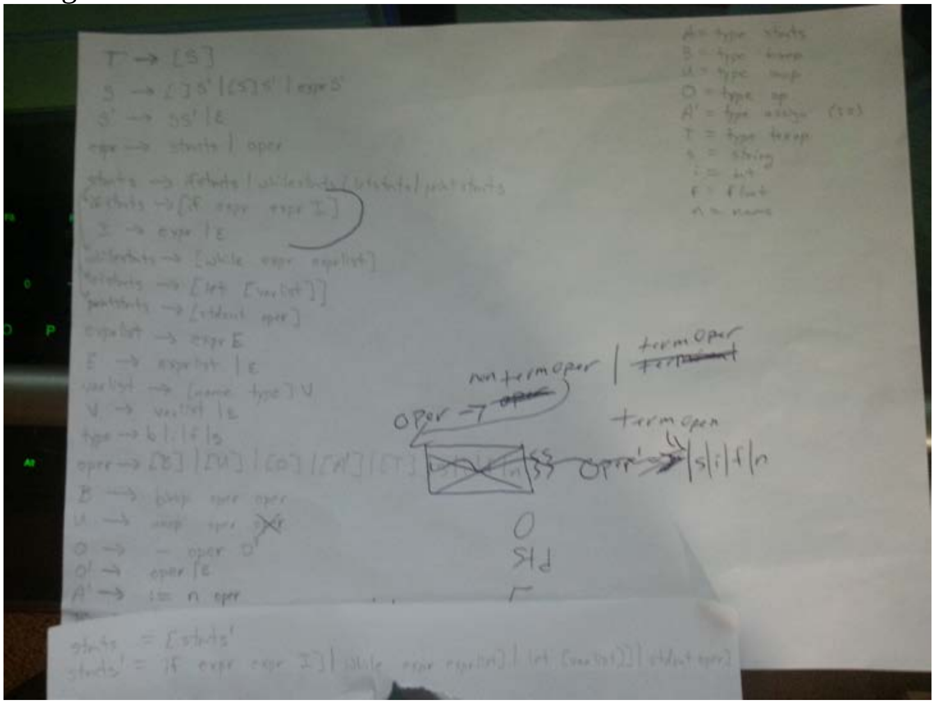
Again these assignments are longer than they appear which is long. We should try developing the tests before the implementation next time, may save time. Forgot to add the floating point with e, but I don't have time to return to that yet, will add in next iteration.

Team Evaluation (what is the percentage of time contributed by each team member)

We were pretty much even this week. Antonio has really been great, my schedule is insane this term and he is being awesome and working around it.

50% Antonio, 50% Me.

Attachments
The grammar:



T → [S]
S → [ ]S' | [S]S' | expr S'
S' → SS' | ε
expr → stmts | oper
stmts → ifstmts | whilestmts | letstmts | printstmts
ifstmts → [if expr expr I]
I → expr | ε
whilestmts → [while expr exprlist]
letstmts → [let [varlist]]
printstmts → [stdout oper]
exprlist → expr E
E → exprlist | ε
varlist → [name type] V
V → varlist | ε
type → b | i | f | s
oper → [B] | [U] | [O] | [A'] | [T] | ~~[illegible]~~ SS
B → binop oper oper
U → unop oper ~~oper~~
O → - oper O'
O' → oper | ε
A' → := n oper

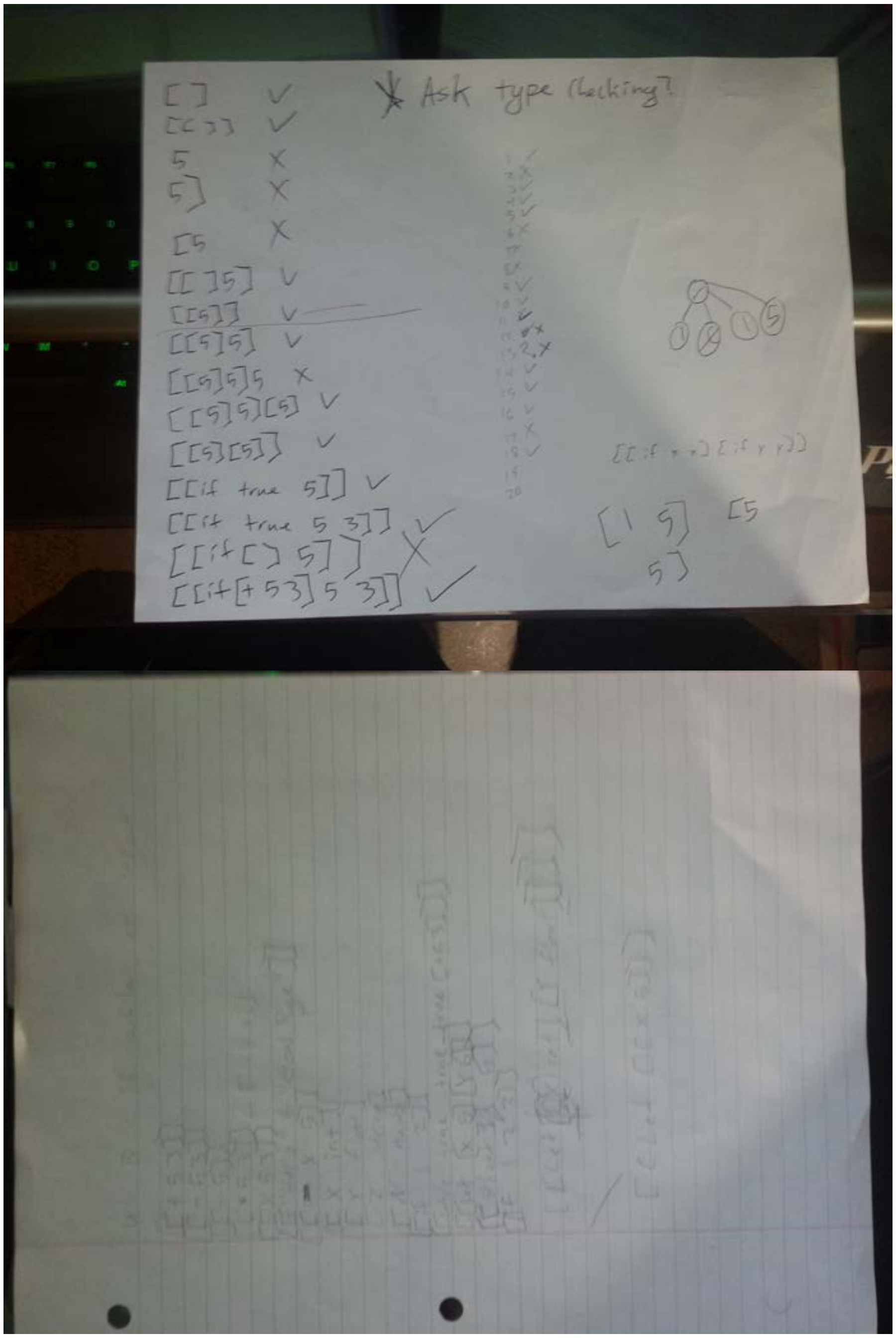
A = type stmts
B = type binop
U = type unop
O = type op
A' = type assign (:=)
T = type [illegible]
s = string
i = int
f = float
n = name

oper → ~~oper~~ nontermoper | termoper ~~[illegible]~~
termopen
oper' → s|i|f|n

stmts = [stmts'
stmts' = if expr expr I] | while expr exprlist] | let [varlist]] | stdout oper]

Name: Riley Hickman
Date: 2/16/2014

Test Cases:



```
[ ]            ✓        Ask type checking?
[[ ]]          ✓
5              X
5]             X
[5             X
[[ ]5]         ✓
[[5]]          ✓
[[5]5]         ✓
[[5]5]5        X
[[5]5][5]      ✓
[[5][5]]       ✓
[[if true 5]]  ✓
[[if true 5 3]]  ✓
[[if [ ] 5] ]    X
[[if [+ 5 3] 5 3]]  ✓
```

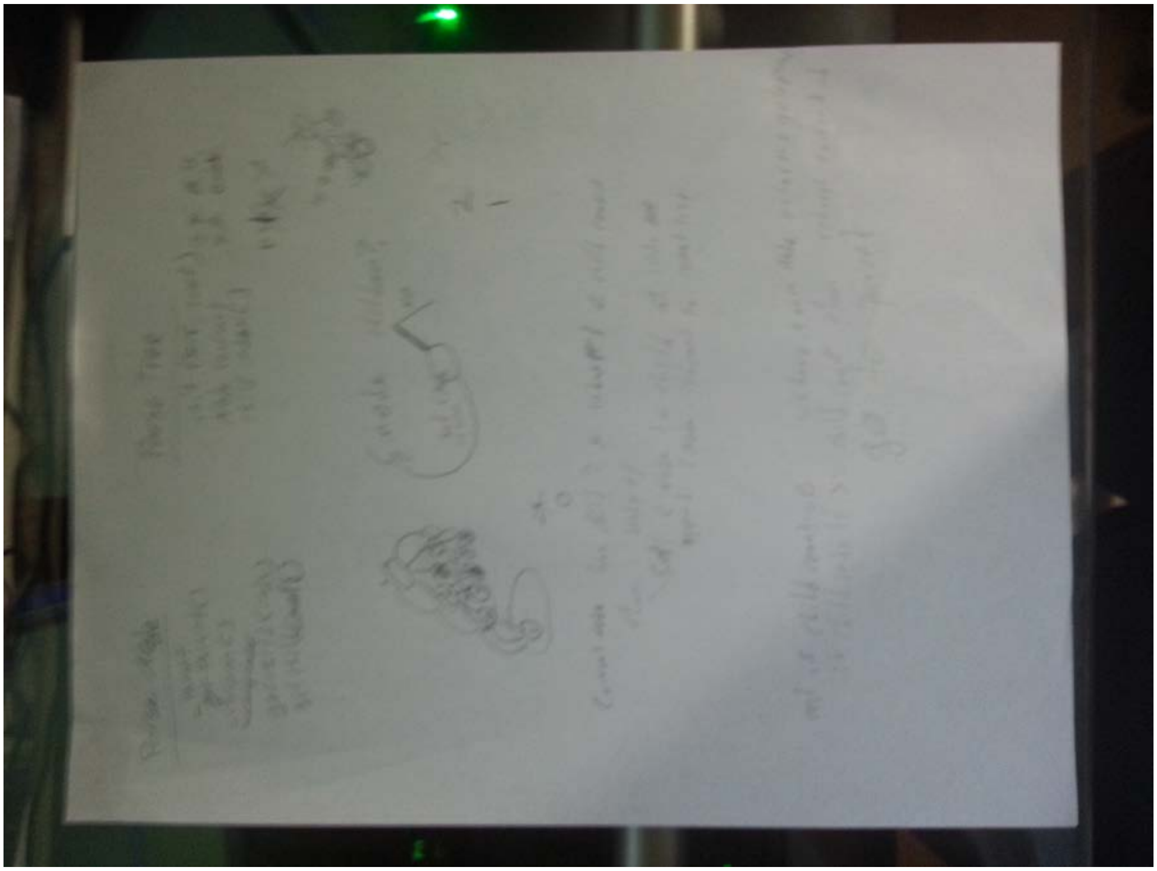
```
1 ✓
2 X
3 ✓
4 ✓
5 ✓
6 X
7
8 X
9 ✓
10 ✓
11 ✓
12 X
13 2,X
14 ✓
15 ✓
16 ✓
17 X
18 ✓
19
20
```

[[if x x] [if y y]]

[1 5]   [5
5]

Name: Riley Hickman
Date: 2/16/2014

ParserTree Notes:



Parse Node

[illegible]

Parse Tree

[illegible]

node children?

[illegible]