Programming Language Concepts Homework 3

Due Wednesday Oct 2; Joseph Sepich (jps6444)

1 Problem 1

1.1 Part 1

```
L(R) = (a|b)(a|b)(a|b)
```

1. “aaa”
2. “aab”
3. “aba”
4. “abb”
5. “baa”
6. “bab”
7. “bba”
8. “bbb”

1.2 Part 2

```
L(R) = a(aa|bb)*b
```

This set will be infinite, so I will write down the 7 shortest.

1. “ab”
2. “aaab”
3. “abbb”
4. “aaaaab”
5. “aaabbb”
6. “abbbbb”
7. “aaaaaaab”

2 Problem 2

2.1 Part 1

Write a regular expression with non empty binaries that start and end with the same digit.

```
L(R) = ((1(1|0)*1)|(0(1|0)*0))
```

2.2 Part 2

Write a regular expression for declartions of variables of type int.

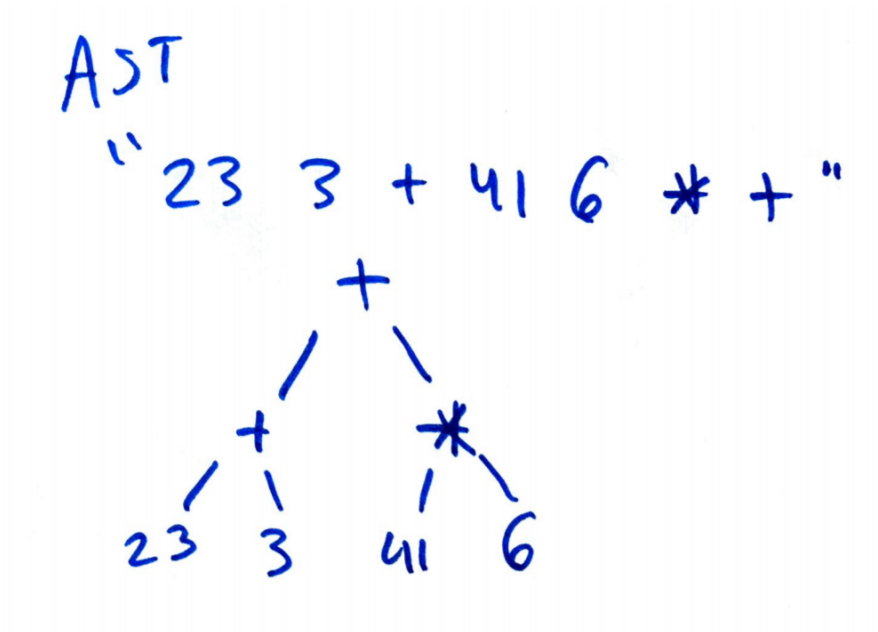
```
L(R) = int\sID(,ID)*=N(,(ID|N))*
```

3 Problem 3

3.1 Part a

Expr -> Expr Expr - | Expr Expr + | Expr Expr * | Expr Expr / | N

3.2 Part b



3.3 Part c

```
Expr -> Expr Expr + -> Expr Expr Expr * + -> Expr Expr N * + ->
Expr N N * + -> Expr Expr + N N * + -> Expr N + N N * + ->
N N + N N * + -> N N + N 6 * + -> N N + 41 6 * + ->
N 3 + 41 6 * + -> 23 3 + 41 6 * +
```

4 Problem 4

Paren -> ϵ | Paren () | () Paren | (Paren)

5 Paroblem 5

5.1 Part a

This language is ambigious. The definition of F has it on either side of an operator. A revised unambigous grammar would be:

```
E -> E + F | F
F -> F * G
G -> Id | (E)
```

This creates a left associativity in the grammar for both the + and * multipliers, and the + multiplier has a lower precedence.

5.2 Part 2

This language is unambigous. The union operator has a lower precendence than the intersenction operator. The union operator has right associativity and the intersection operator has a left associativity.