1. Discuss adding ++ and -- operators to Clite. Assume they are allowed only as free standing statements.

(a) Discuss the changes required to the type checker and type transformer

Adding ++ and -- involves modifying the type checker to recognize ++ as increment or decrement (i=i+1 or i=i-1) as a binary expression that maps and alters i's state in the hashmap every time its envoked. The type transformer would need to be modified to recognize a in/decrementing statement according to whether or not i was an int or float. int values in a float context would need to be coerced accordingly.

(b) An in/decrement can be defined for the prefix or suffix of an assignments identifier, ie. ++i, i++, --i, i--. Apply a binary expression evaluation that assigns a provided identifier a +1 or -1.

(c) Assignment applyIncr (Variable v)

* evaluate v

* add 1 to v

* return assignment w/ new value (+1)

```
Value localexpr = new IntValue(v.intValue() + 1);
   return new Assignment(v, localexpr);
```

Assignment applydec (Variable v)

* eval v

* sub 1 to v

* return assignment w/ new value (-1)

```
Value localexpr = new IntValue( v.intValue()-1);
   return new Assignment (v, localexpr);
```

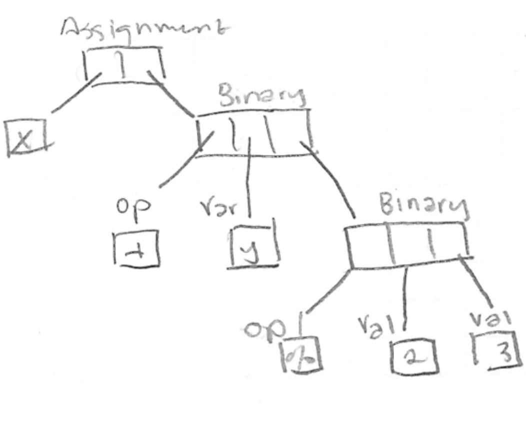
(d) Having to adjust definitions of the type transformer and static type checker on top of conflating the meaning of an expression can be solved by using assignment statements explicitly i=i+1, i=i-1.

4. Add the operator % to Clite

(a) – (d) We can add the % operator by adding '%' to the concrete syntax of Clite, MulOp → * | / | %

An example of an abstract syntax tree for % might look like
Operator = + | - | * | / | ! | %

x = y + 2%3



add to Token.java

public static final Token modTok = new
    (Token(TokenType.Modulo, "%")

add to TokenType.java

{ ... Minus, Multiply, Modulo, ... }

add to Lexer.java

```
public Token next()
  ... case '%':
      ch = nextChar();
      return Token.modTok;
```

8.
```
1  void main() {
2      int i, a, z;
3      i = 5;
4      a = 2;
5      z = 1;
6      while (i > 0) {
7          if (i - i / 2 * 2 == 1) ✓ ∅
8              z = z * a;          z: 2, 2
9          i = i / 2;              i: 2, 1
10         a = a * a;              a: 4, 16
11     }
12 }
```

- i

*

Step	Before statement	i	a	z
		Variables		
1	3	undef	undef	undef
2	4	5	undef	undef
3	5	5	2	undef
4	6	5	2	1
5	7	5	2	1
6	8	5	2	1
7	9	5	2	2
8	10	2	2	2
9	6	2	4	2
10	7	2	4	2
11	8	2	4	2
12	9	2	4	2
13	10	1	4	2
14	6	1	16	2
15	7	1	16	2
16	8	1	16	2
17	9	1	16	2
18	10	0	16	2
19	6	0	256	32
20	12	0	256	32

9.

Parser
```
Assignment:
    Variable: z
    Binary:
        Operator: *
        Variable: z
        Variable: a
```

Type Transformer
```
Assignment:
    Variable: z
    Binary:
        Operator: INT*
        Variable: z
        Variable: a
```

c. Using Expression semantic Meaning rules, the evaluation of variables are defined by their mapping environment. By Meaning Rule 8.7.2, If the expression is a variable, then its meaning is the value of the variable in the current state.

```
Value M(Expression e, State state) {
    ...
    if (e instanceof Veriable)
        return (Value) (state.get(e));

applyBinary (...) { ...

    if (op.val.equals(Operator.INT_TIMES))
        return new IntValue(
            v1.intValue() * v2.intValue());
```

(a) $M(x+2*y, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
(b) $M(2*x+3/y-4), \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
(c) $M(1, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$

M. Expression x State → Value

a – $M(\underbrace{(x+2)}_{}*\underbrace{y}_{}, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\}) = -4$

Binary Expression MR 8.7:3 → 8.8

$M((x+2)*y, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\}) =$
ApplyBinary $(+, A, B)$ where $A = M(x, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
$B = M(2*y, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
$A = 2$

$B = M(\underbrace{2*y}_{}, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$

Binary Expression MR 8.7:3 → 8.8
B. ApplyBinary $(*, C, D)$ where $C = M(2, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
$D = M(y, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
$C = 2,\ D = -3$

$\Rightarrow B = M(2*y, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\}) =$ ApplyBinary $(*, 2, -3)$
$= -6$

$\Rightarrow M(x+2*y, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\}) =$ Apply Binary $(+, 2, -6)$
$= -4$

---

b – $M(\underbrace{2*x+3/y}_{}\,-\underbrace{4}_{}, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$

Binary Expression MR 8.7:3 → 8.8
ApplyBinary $(-, A, B)$ $A = M(2*x+3/y, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
$B = M(4, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
A. ApplyBinary $(+, C, D)$ $C = M(2*x, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
$D = M(3/y, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
C. ApplyBinary $(*, E, F)$ $E = M(2, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
$F = M(x, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
$F = M(2, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
D. ApplyBinary $(/, G, H)$ $G = M(3, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
$H = M(y, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$
$H = M(-3, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\})$

$\Rightarrow -1$

c – $M(1, \{\langle x,2\rangle, \langle y,-3\rangle, \langle z,75\rangle\}) = 1$

Given Meening rule 8.7:1