Homework 9

Return by: November 25

Q1. What is the smallest well-typed FJ program that would fail to type check if we removed:

a) T <: T (reflexivity)
b) T <: T" iff T<:T' and T'<:T" (transitivity)

For (a) you can assume an empty CT. For (b) we can assume a CT with two classes A, B. (No need to write the code of those).

Q2. Is FJ deterministic? (Don’t worry about parsing) `Answer:    yes / no`

Q3. How many static errors (i.e. statements that prevent the OK rule to hold) are there in the following FJ program (assuming a CT with the definition of Pair as in the paper)

```
class Pear extends Pair {
    Object flavor;
    Pear(Object fst, Object snd, Object flavor){super(snd,fst); this.flavor=flavor;}
    Pear  setfst(Object f) { return new Pear(this,this,this); }
}
```

```
Answer:    0 / 1 / 2 / 3 / 4
```

Q4) How many static errors are there in the following FJ program (assuming a CT with the definition of Pair as in the paper)

```
class Triple extends Pair {
    Object third;
    Triple(Object fst, Object snd, Object third){super(fst,snd); this.third=third;}
    Triple weird() {return new Triple(this,this,this);}
    Triple change() {return (Triple)new Pair(new Object(),new Object()); }
}
```

```
Answer:    0 / 1 / 2 / 3 / 4
```

Q5) Given a definition of structural subtyping where T <:struct T' iff any field/method occuring in T' also occur T and the type signature of these fields/methods are identical. Is it always the the case that if T<:T' then T<:struct T' holds.

```
Answer: yes / no
```

Q6) Write all the reduction steps in the evaluation of this FJ program:

```
    new Triple(new Object(), new Object(), new Object()).weird().setfst(new Object())
```

Q7) Add mutable state to FJ and prove type preservation.