Exercise 11

Exercise 1

Recall from the lecture that the pointer analysis abstract domain is defined as:

$$\mathsf{Labs} \to ((\mathsf{PtrVar} \to \mathcal{P}(\mathsf{AbsObj})) \times (\mathsf{AbsObj} \times \mathsf{Field} \to \mathcal{P}(\mathsf{AbsObj})))$$

, where the abstract domain keeps two maps at every program label. The first map contains a mapping from a pointer variable to a set of abstract objects. The second map contains a mapping from the fields of abstract objects to the set of abstract objects they point to.

- Write down the formal definition for all the abstract transformers that capture the effect of program statements manipulating pointers on the abstract domain:

(object creation)	$p := newObject^l$
(compare two pointers)	$p = q$
(pointer assignment)	$p := q$
(pointer heap store)	$p.f := q$
(pointer heap load)	$p := q.f$

- Define:
  1. the partial order $\sqsubseteq$
  2. the least ($\bot$) and greatest ($\top$) elements
  3. the meet $\sqcap$
  4. the join $\sqcup$

Exercise 2

You are given the following program:

```
0:  c = newObject T;
1:  t = c;
2:  i = 0;
```

```
3:  while (i < count) {
4:    n = newObject T;
5:    c.f = n;
6:    c = n;
7:    i++;
8:  }
9:  c.f = t;
10: assert t != n;
```

1) Run the flow-sensitive pointer analysis from the lecture on it.
2) Can you prove the assertion on line 10 using the results of the analysis?

Exercise 3

Write a program for which the flow-sensitive pointer analysis from the lecture infers the following abstract state at the end of the program:

```
{ a->{A0}, b->{A0,A1}, A0.f->{A0}, A1.f->{A0} }
```

Exercise 4

Run both the flow-sensitive and the flow-insensitive pointer analysis on the following program:

```
0: a = newObject T;
1: b = a;
2: if (a == b) {
3:   b = newObject T;
4: } else {
5: }
```