Exercise 11 (Solution)

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)	$p := q.f$

  Solution: Let (m_p, m_h) be a tuple where m_p is a pointer map and m_h is a heap map, pointed to by some label l. The notation $m[x \mapsto y]$ denotes the map m modified to yield y when applied to x.

  Object creation:

  $[\![p := newObject^l]\!](m_p, m_h) = (m_p[p \mapsto l], m_h)$

  Comparing two pointers:

  $[\![p = q]\!](m_p, m_h) = (m_p[p \mapsto m_p(p) \cap m_p(q), q \mapsto m_p(p) \cap m_p(q)], m_h)$

  Pointer assignment:

  $[\![p := q]\!](m_p, m_h) = (m_p[p \mapsto m_p(q)], m_h)$

Pointer heap store (weak-update):

$[\![p.f := q]\!](m_p, m_h) = (m_p, m_h[a.f \mapsto m_h(a.f) \cup m_p(q) \mid a \in m_p(p)])$

Pointer heap load:

$[\![p := q.f]\!](m_p, m_h) = (m_p[p \mapsto \bigcup_{x \in m_p(q)} m_h(x.f)], m_h)$

- Define:
  1. the partial order $\sqsubseteq$

     Solution:

     $((m_{p1}, m_{h1}) \sqsubseteq (m_{p2}, m_{h2}))(l) \leftrightarrow \lambda p.\ m_{p1}[p](l) \subseteq m_{p2}[p](l) \wedge \lambda(a.f).\ m_{h1}[a.f](l) \subseteq m_{h2}[a.f](l)$

  2. the least ($\bot$) and greatest ($\top$) elements

     Solution:

     $\bot(l) = (m_p^\bot, m_h^\bot)$ where $\begin{cases} m_p^\bot = \lambda p.\ \emptyset \\ m_h^\bot = \lambda(a.f).\ \emptyset \end{cases}$

     $\top(l) = (m_p^\top, m_h^\top)$ where $\begin{cases} m_p^\top = \lambda p.\ \mathsf{AbsObj} \\ m_h^\top = \lambda(a.f).\ \mathsf{AbsObj} \end{cases}$

  3. the meet $\sqcap$

     Solution:

$((m_{p1}, m_{h1}) \sqcap (m_{p2}, m_{h2}))(l) = (m_p, m_h)$ where $\begin{cases} m_p = \lambda p.\ m_{p1}[p](l) \cap m_{p2}[p](l) \\ m_h = \lambda(a.f).\ m_{h1}[a.f](l) \cap m_{h2}[a.f](l) \end{cases}$

  4. the join $\sqcup$

     Solution:

$((m_{p1}, m_{h1}) \sqcup (m_{p2}, m_{h2}))(l) = (m_p, m_h)$ where $\begin{cases} m_p = \lambda p.\ m_{p1}[p](l) \cup m_{p2}[p](l) \\ m_h = \lambda(a.f).\ m_{h1}[a.f](l) \cup m_{h2}[a.f](l) \end{cases}$

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.
Solution:

```
0:  c = newObject T;
    { t->{}, c->{A0}, n->{} }
1:  t = c;
    { t->{A0}, c->{A0}, n->{} }
2:  i = 0;
    { t->{A0}, c->{A0}, n->{} }
3:  while (i < count) {
      { t->{A0}, c->{A0, A4}, n->{A4}, A0.f->{A4}, A4.f->{A4} }
4:    n = newObject T;
      { t->{A0}, c->{A0, A4}, n->{A4}, A0.f->{A4}, A4.f->{A4} }
5:    c.f = n;
      { t->{A0}, c->{A0, A4}, n->{A4}, A0.f->{A4}, A4.f->{A4} }
6:    c = n;
      { t->{A0}, c->{A4}, n->{A4}, A0.f->{A4}, A4.f->{A4} }
7:    i++;
      { t->{A0}, c->{A4}, n->{A4}, A0.f->{A4}, A4.f->{A4} }
8:  }
    { t->{A0}, c->{A0,A4}, n->{A4}, A0.f->{A4}, A4.f->{A4} }
9:  c.f = t;
    { t->{A0}, c->{A0,A4}, n->{A4}, A0.f->{A0,A4}, A4.f->{A0,A4} }
10: assert t != n;
```

2) Can you prove the assertion on line 10 using the results of the analysis?
Solution: No, since variables `t` and `n` could be both `null`.

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} }
```

Solution:

```
0: a = newObject T;
   { a->{A0} }
1: b = newObject T;
   { a->{A0}, b->{A1} }
```

```
2: if (*) {
     { a->{A0}, b->{A1} }
3:   b = a;
     { a->{A0}, b->{A0} }
4: }
   { a->{A0}, b->{A0,A1} }
5: b.f = a;
   { 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: }
```

Solution:
1) Flow-sensitive pointer analysis:

```
0: a = newObject T;
   { a->{A0} }
1: b = a;
   { a->{A0}, b->{A0} }
2: if (a == b) {
     { a->{A0}, b->{A0} }
3:   b = newObject T;
     { a->{A0}, b->{A3} }
4: } else {
     { a->{A0}, b->{A0} }
5: }
   { a->{A0}, b->{A0,A3} }
```

2) Flow-insensitive pointer analysis:

```
0: a = newObject T;
1: b = a;
2: if (a == b) {
3:   b = newObject T;
4: } else {
5: }
   { a->{A0}, b->{A0,A3} }
```