1. Consider the following first implementation of the lockset algorithm:

   Let $locks\ held(t)$ be the set of locks held by thread $t$.

   For each $v$, initialize $C(v)$ to the set of all locks.
   On each access to $v$ by thread $t$,
   set $C(v) := C(v) \cap locks\ held(t)$;
   if $C(v) = \{\}$, then issue a warning.

For the following example program execution, show the values of locks\_held and $C(A)$ for variable A over time. Will a data race detected? If so, when?

```
Lock(i)
Lock(m);
A++;  [locks\_held: i, m]
Unlock(m);

Lock(m);
A--;  [locks\_held: i, m]
Unlock(m);
Unlock(i);

Lock(i);
Lock(j);
A++;  [locks\_held: i, j]
Unlock(i);
Unlock(j);

Lock(i);
B++;  [locks\_held: i]
Unlock(i);

Lock(m);
A--;  [locks\_held: m]
Unlock(m);
```

(Optinal: Can you come up with a better code example?)
2. Consider the following new states for a variable to handle initialization and read-sharing. How is C(v) updated when a variable v is in each state? What checks are made for a variable in each state?

For the following code sequence, what is the state of each variable A and B over time? Over time, what are the values of locks_held and C(A) and C(B)? Would a race be reported? Can any of these locks be removed and still be valid?

### Thread 1

```plaintext
A, B virgin
// code not involving A or B
A = 1;
B = A+5;
A++; Excl.
Lock(n);
B++; Excl.
Unlock(n);
C(B) = \( \frac{3}{2} n^2 \)
```

### Thread 2

```plaintext
A, B virgin
// code not involving A or B;
Lock(n);
B--; Excl.
Unlock(n);
C(B) = \( \frac{3}{2} n^2 \)
Lock(m);
If (A == 2) {
    // do something();
} Unlock(m);
C(A) = \( \frac{3}{2} m^2 \)
```

(Optional: Can you come up with a better code example?)
3. Consider the following version to handle read-write locks:

Let \( \text{locks held}(t) \) be the set of locks held in any mode by thread \( t \).
Let \( \text{write locks held}(t) \) be the set of locks held in write mode by thread \( t \).
For each \( v \), initialize \( C(v) \) to the set of all locks.
On each read of \( v \) by thread \( t \),
set \( C(v) := C(v) \cap \text{locks held}(t) \);
if \( C(v) = \{\} \), then issue a warning.
On each write of \( v \) by thread \( t \),
set \( C(v) := C(v) \cap \text{write locks held}(t) \);
if \( C(v) = \{\} \), then issue a warning.

For the following code sequence, what is the state of each variable \( A \) and \( B \) over time? Over time, what are the values of \( \text{locks held} \), \( \text{write locks held} \), and \( C(A) \) and \( C(B) \)? Would any races be reported (if so, where)? Can any of these locks be removed and still be valid? Assume if Lock is given the parameter \text{WRITER} \ it is a write lock, otherwise it is only a read lock.

(Optional: Can you come up with a better code example?)

<table>
<thead>
<tr>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A = 1; B = A+5; C = B+1;</td>
<td>Lock(k); Lock(j); If (A == ...) do something(); Unlock(j); Unlock(k);</td>
<td>Lock(e); Lock(g); If (C == ...) do something(); Unlock(g); Unlock(e);</td>
</tr>
<tr>
<td>Lock(k); Lock(j); Unlock(j); Unlock(k);</td>
<td>C(A) = ∅, ∅ biodiversity</td>
<td>Lock(m); Lock(n); If (A == ...) do something(); Unlock(n); Unlock(m);</td>
</tr>
<tr>
<td>Lock(d); If (B == ...) do something(); Unlock(d);</td>
<td>C(c) = ∅ biodiversity</td>
<td>Lock(g, WRITER) Lock(d); B++; Unlock(d); Unlock(g);</td>
</tr>
<tr>
<td>Lock(e, WRITER); Lock(c, WRITER); C++; Unlock(c); Unlock(e);</td>
<td>C(c) = ∅ biodiversity</td>
<td>C(B) = NULL! d needed to be writer lock!!</td>
</tr>
<tr>
<td>C-OKAY</td>
<td>C(A) = NULL biodiversity but no warning</td>
<td>null.. but no warning</td>
</tr>
</tbody>
</table>

A = no common lock out taken if only read