

## Solutions to Exercise 8

**Problem 1.**

- Figure 1. Yes. An equivalent serial execution is  $T_2 \cdot T_1$ .
- Figure 2. Yes. An equivalent serial execution is  $T_2 \cdot T_1$ .
- Figure 3. Yes. An equivalent serial execution is  $T_2 \cdot T_1$ .
- Figure 4. No. The execution is not opaque because  $T_3$  observes results of  $T_1$ 's actions even though  $T_1$  is aborted. One way to make it opaque is to have the read operations in  $T_3$  return 0. In this case an equivalent sequential execution is  $T_1 \cdot T_3 \cdot T_2$ .
- Figure 5. No. The execution is not opaque because if  $T_1$  is serialized before  $T_2$ , then  $T_2$  does not observe the write to  $y$ ; and if  $T_2$  is serialized before  $T_1$ , then  $T_1$  does not observe the write to  $x$ . One way to make the execution opaque is to abort one of the transactions. Another is to have read operation in  $T_1$  return 1. In this case an equivalent serial execution is  $T_2 \cdot T_1$ .
- Figure 6. Yes. An equivalent sequential execution is  $T_1 \cdot T_2$ .

**Problem 2.** To implement these objects using transactional memory, we only need to enclose their sequential specification in an atomic block.

Snapshot:

**uses:**  $array[M]$

**upon Snapshot do**

```
begintransaction;  
for  $i = 1$  to  $M$  do  
   $ret[i] \leftarrow array[i]$ ;  
endtransaction;  
return  $ret$ 
```

Counter:

**initially:**  $count = 0$

**upon Inc do**

```
begintransaction;  
 $ret \leftarrow count$ ;  
 $count \leftarrow count + 1$ ;  
endtransaction;  
return  $ret$ 
```

CASN:

**uses:**  $array[M]$

**upon CASN( $idx, oldv, newv$ ) do**

```
begintransaction;  
 $L \leftarrow length(idx)$ ;  
for  $i = 1$  to  $L$  do  
  if  $array[idx[i]] \neq oldv[i]$  then  
    endtransaction;  
    return  $array$   
for  $i = 1$  to  $L$  do  
   $array[idx[i]] \leftarrow newv[i]$   
endtransaction;  
return  $array$ 
```