Strict Timestamp Ordering Algorithm

(Guarantees timestamp order of transaction execution,
recoverability, and conflict serializability)

Each active database item will have the following two timestamps:

read_TS(X): the sequence number of the youngest transaction that read item X.
write_TS(X): the sequence number of the youngest transaction that wrote item X.

Every transaction is assigned with a unique integer sequence number.
$TS(T_9)$ is the timestamp of T_9 which is the sequence number of the transaction = 9

Consider two transactions T_5 and T_8, where T_8 has started after T_5.
Hence, T_8 is younger to T_5. Also, $TS(T_8) > TS(T_5)$.

T_9 requests write_item(X):

```
If ((read_TS(X) > TS(T9)) OR (write_TS(X) > TS(T9)))
{
     T9 will abort; //some younger trans. has read/written X
}
Else
{
     T9 will wait till the youngest transaction (Twrite_TS(X))
        that has written X, is committed/aborted.
     T9 performs write_item(X);
     write_TS(X) = TS(T9);
}
```

T_9 requests read_item(X):

```
If (write_TS(X) > TS(T9))
{
     T9 will abort; //some younger trans. has written X
}
Else
{
     T9 will wait till the youngest transaction (Twrite_TS(X))
          that has written X, is committed/aborted.
     T9 performs read_item(X);
     read_TS(X) = max(read_TS(X),TS(T9));
}
```