Exercise (Recovery algorithm)

Apply the recovery algorithm executed by the system when recovers from a crash.
Assume the following log:

<T1 start> <T2 start> <T2, O1, V1, V1'> <T1, O2, V2, V2'> <T3 start> <T1 commit> <T4 start> <T3, O3, V3, V3'> <T4, O4, V4, V4'> checkpoint (L) <T5 start> <T4 commit> <T3, O5, V5, V5'> <T5, O6, V6, V6'> <T3, O7, V7, V7'><T3 commit> <T5 commit> <T2, O8,V8,V8'> CRASH

with
<T, O1, V1, V1'> record for update operation : transaction T updates object O1;
V1 is the state of O1 before the update; V1' is the state of O1 after the update.

1) Shows the list L of transactions active at the checkpoint.
2) Show the undo-list and the redo-list.
3) Show the actions executed by the system in the correct order.

Point 1

L= {T2, T3, T4}

Point 2

undo-list ={} redo-list ={}	
undo-list ={} redo-list ={ T5}	<commit, T5>
undo-list ={} redo-list ={T3, T5}	<commit, T3>
undo-list ={} redo-list ={T3, T4, T5}	<commit, T4>
undo-list ={T2 } redo-list ={T3,T4, T5}	ckeckpoint(T2, T3, T4)

Point 3

3.1) Rescan the log backward and perform undo for each log record that belongs to Ti in undo_list. Log records that belong to transactions in redo_list are ignored. The scan stops when the <Ti start> records have been found for every transaction in undo_list.

Undo actions:
T2: O6 = V6
T2: O1 = V1

3.2) Scan the log backward until the <start, Ti> records have been found for every transaction in redo_list. Scan the log forward. For each record of transactions in redo_list perform redo of the operation.

Redo actions:
T3:
T4: O3=V4
T3:
T5: O4=V6
T3: