CSE-5331
DBMS MODELS AND IMPLEMENTATION
ASSIGNMENT 1 – FINAL REPORT

Team Members:

Sai Rohith Pasala(1001873156) <--------------------ONLY THIS STUDENT SUBMITS THE DOCUMENT
Thirumanyam Tushar Kanth Reddy(1001859165)

All the work is done together and equally.

Design:
We decided to implement this project in python programming language using 're' library and 'sys' library.

Programming language Used: Python.
Text Editor Used: Notepad++, Notepad.
Data Structures/ Functions Used: Lists, List of Lists(table), String functions, Real expression functions, System Argument functions.

An algorithm for cautious waiting was proposed to avoid needless aborts or restarts. The transaction Ti
tries to lock item X but is unable to do so because X is locked by another transaction Tj with a conflicting lock. Here are the cautious waiting rules:

- **Cautious Waiting**: If Tj is not blocked (not waiting for some other locked item), then Ti is Blocked and allowed to wait; otherwise abort Ti

Case 1: If Ti < Tj: Abort Ti and give the resource to Tj.
Case 2: If Ti > Tj: Execute Ti and put Tj in the waiting queue.

- **Wait Die**: When an older transaction tries to lock a DB element that has been locked by a younger transaction, it waits. When a younger transaction tries to lock a DB element that has been locked by an older transaction, it dies.

Case 1: If Ti < Tj: Tj is executed first and Ti is allowed to wait until the next available data item.

Case 2: If Ti>Tj: Ti dies and is restarted later. Tj is executed first.

We can prove that cautious waiting is deadlock-free, because no transaction will ever wait for another blocked transaction. By considering the time b(T) at which each blocked transaction (T) was blocked, if the two transactions Ti and Tj above both become blocked, and Ti is waiting for Tj , then b(Ti) < b(Tj), since,
Ti can only wait for Tj at a time when Tj is not blocked itself. The blocking times form a total ordering of all blocked transactions, so no deadlock can occur.

1. Transaction Table (list)
    a. Blockedby
    b. Blocked operations
    c. Timestamp
    d. State
    e. TID

                f) Operation Methods:
                    i) Change state
                    ii) Clear locks
                    iii) Lock item

2) Locks Table (list)
    a) TIDs list
    b) Data item
    c) Lock mode
    d) State
    e) Blocked TIDs
    f) Operation Methods:
        i) Append TID
        ii) Append Blocked TIDs

Inbuilt data structure like Python dictionaries use of following, instead of classes,

- Lock Table (list)
- Transactions Table (list)
- Blocked items Table (list)

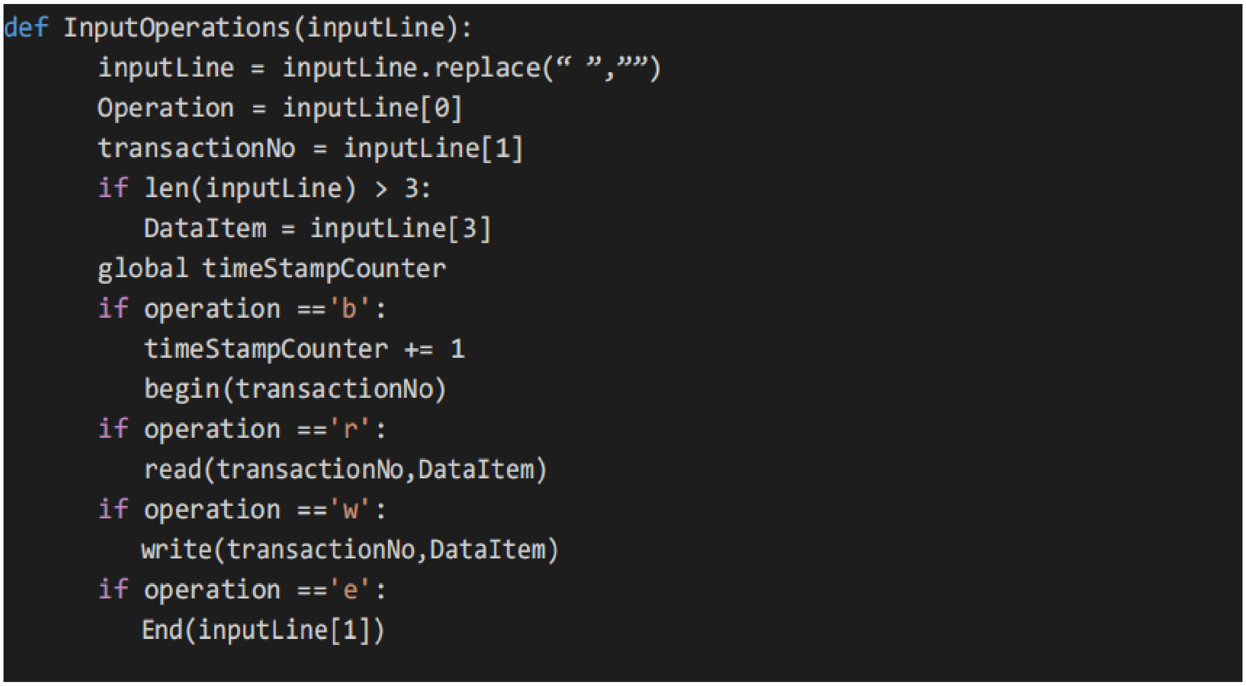
Methods/utilities:

- unlock()
- readlock (tid, data item)
- writelock (tid, data item)
- read(tid)
- write(tid)
- abort(tid)
- begin(tid)
- input operations(commands)
- blocked(tid)
- deadlock(tid,data item,mode)

Input operations function:

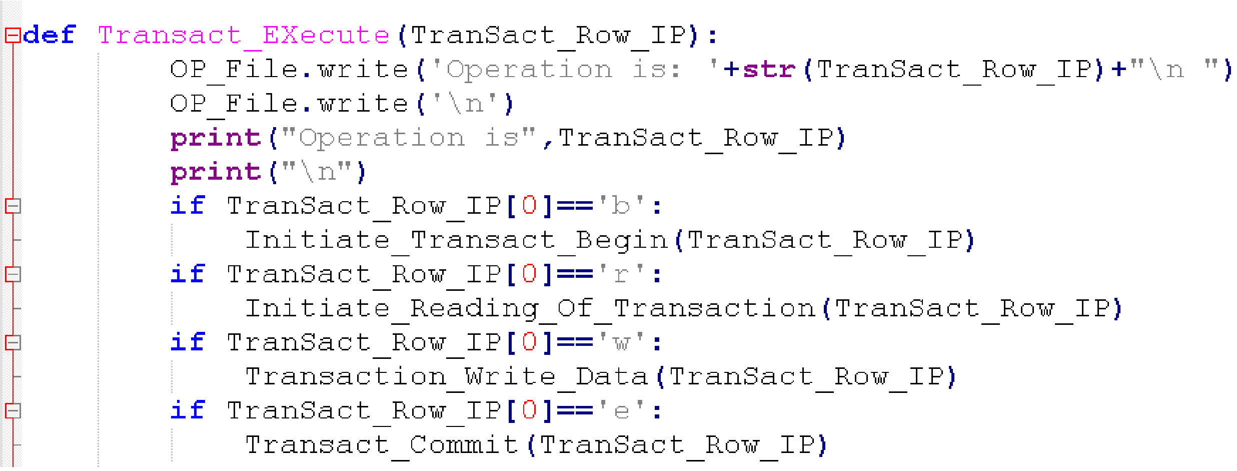
1) Read input line
2) Begin transaction
    a) Mark transaction as Active
3) Read operation
    a) Acquire read lock on object
4) Write operation
    a) Acquire write lock on object
5) End transaction
    a) Release all locks

Pseudo Code(Sample):



```
def InputOperations(inputLine):
      inputLine = inputLine.replace(“ ”,””)
      Operation = inputLine[0]
      transactionNo = inputLine[1]
      if len(inputLine) > 3:
         DataItem = inputLine[3]
      global timeStampCounter
      if operation =='b':
         timeStampCounter += 1
         begin(transactionNo)
      if operation =='r':
         read(transactionNo,DataItem)
      if operation =='w':
         write(transactionNo,DataItem)
      if operation =='e':
         End(inputLine[1])
```

Sample Code used for above Pseudo Code:



```
def Transact_EXecute(TranSact_Row_IP):
        OP_File.write('Operation is: '+str(TranSact_Row_IP)+"\n ")
        OP_File.write('\n')
        print("Operation is",TranSact_Row_IP)
        print("\n")
        if TranSact_Row_IP[0]=='b':
            Initiate_Transact_Begin(TranSact_Row_IP)
        if TranSact_Row_IP[0]=='r':
            Initiate_Reading_Of_Transaction(TranSact_Row_IP)
        if TranSact_Row_IP[0]=='w':
            Transaction_Write_Data(TranSact_Row_IP)
        if TranSact_Row_IP[0]=='e':
            Transact_Commit(TranSact_Row_IP)
```

Example 1:

B1; # calls **begin** transaction TID=T1 TS=1

R1(Y); # acquires **read lock** on Y for TID T1

W1(Y); # acquires/upgrades **write lock** on Y for TID T1

R1(Z); # acquires **read lock** on Z for TID T1

B3; # calls **begin** transaction TID=T3 TS=2

R3(X); # acquires **read lock** on X for TID T3

W3(X); # acquires/upgrades **write lock** on X for TID T1W1(Z);

# acquires/upgrades **write lock** on Z for TID T3

E1; # **unlock** TID=T1 release lock on Y,Z resume **waiting operations**

E3; # **unlock** TID=T3 release lock on X,Z resume **waiting operations**

Example 2:

B1; # calls **begin** transaction TID=T1 TS=1

R1(Y); # acquires **read lock** on Y for TID T1

W1(Y); # upgrades **write lock** on Y for TID T1

R1(Z) # acquires **read lock** on Z for TID T1

B2 # calls **begin** transaction TID=T2 TS=2

R2(Y) # deadlock encountered, the R2(Y) gets **blocked by** Transaction T1

B3; calls **begin** transaction TID=T3 TS=3

R3(Z); # acquires **read lock** on Z for TID T3(Item Z read locked by

T3)

W3(Z); Deadlock is Encountered, the W1(Z) gets **blocked by**

Transaction T1

E1; # **unlock** TID=T1 **release lock** on Y,Z resume **waiting operations** for items

W3(Z); upgrades **read lock** to write lock for item z

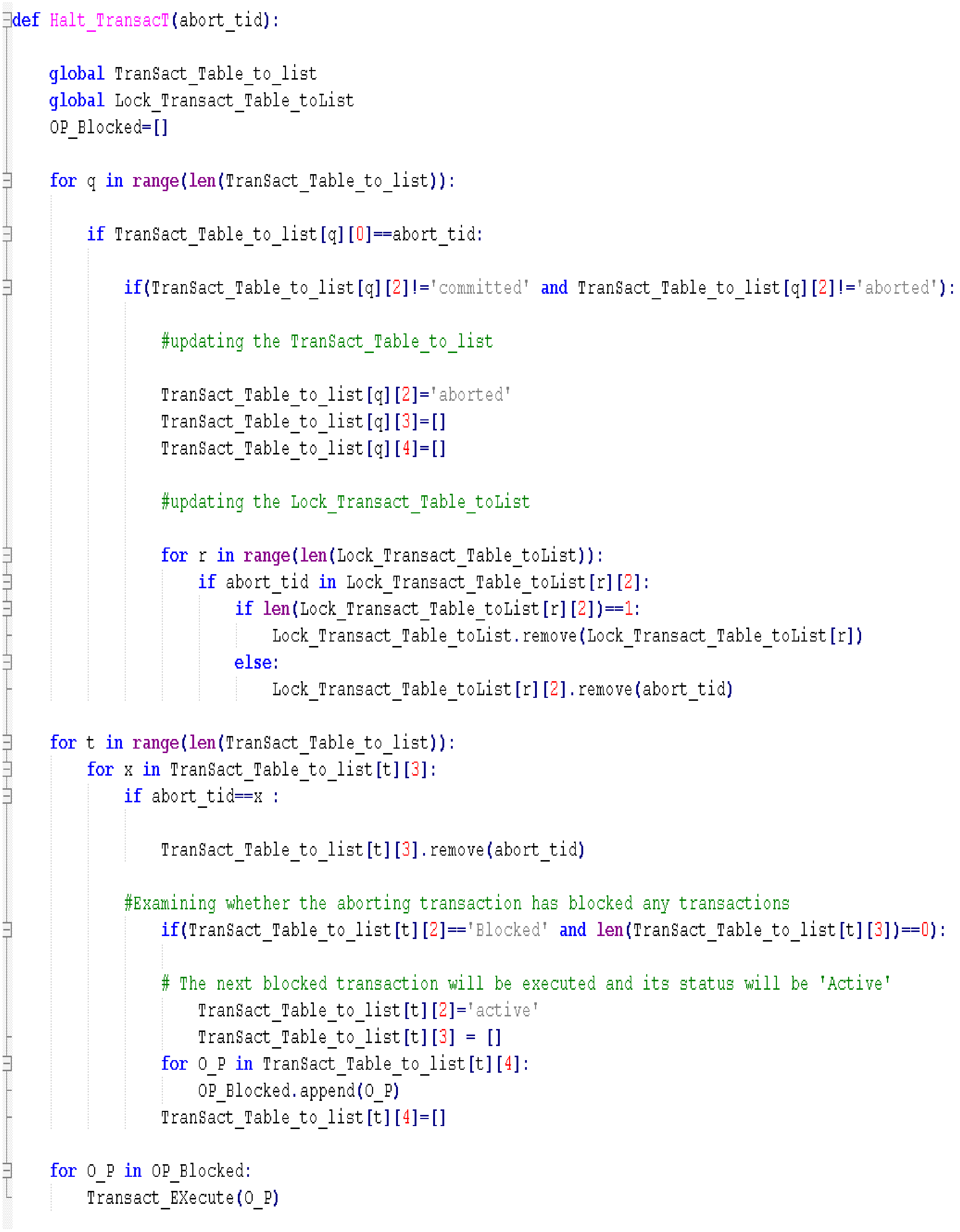
E3; # **unlock** TID=T3 **release lock** on X,Z resume **waiting operations** for items

Difference between cautious wait and wait die in implementation:

The main difference we can see is :

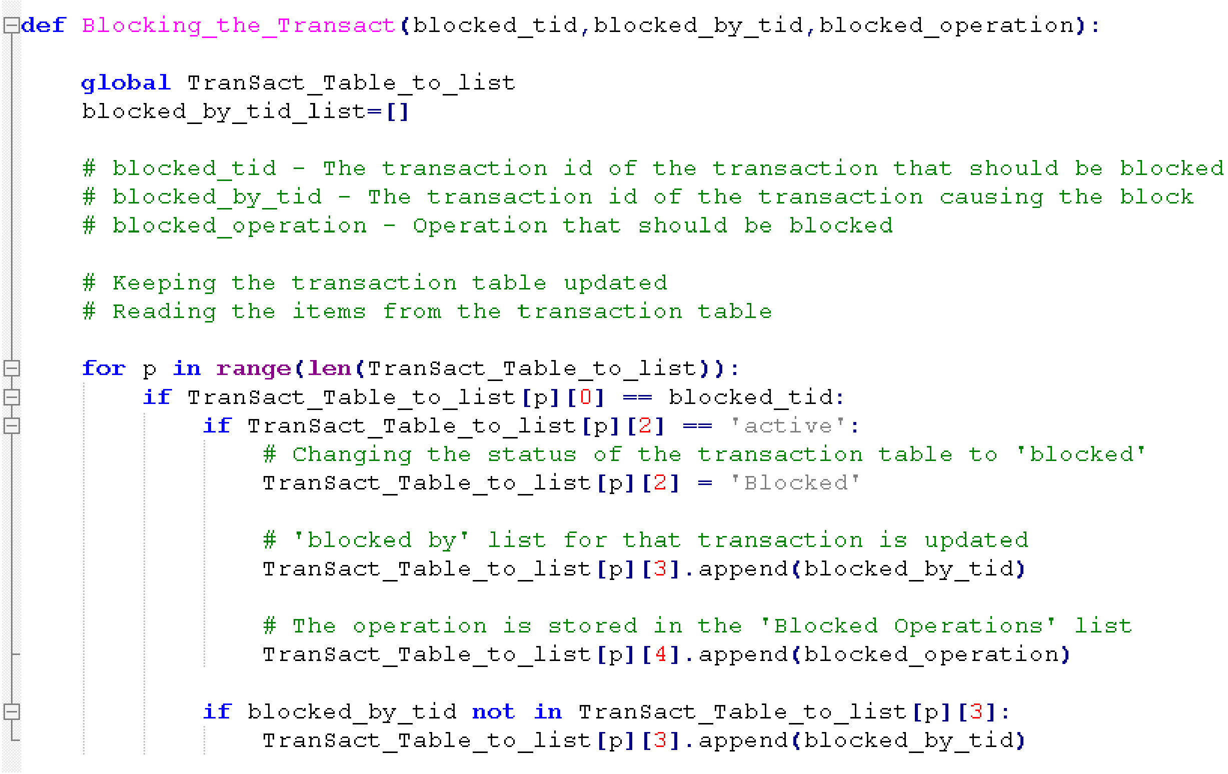
- Timestamp comparison in wait die whereas it is not there in cautious wait.
- Abort function is slightly changed for both the protocols appropriately.
- Block function is slightly changed for both the protocols appropriately.
- Write function is slightly changed for both the protocols appropriately.
- Check in Read Function which is used check whether the item is trying to lock onto a previously locked item in the transaction is there for both with different implementation.

CODE FOR CAUTIOUS-WAIT PROTOCOL: (1) Abort Function



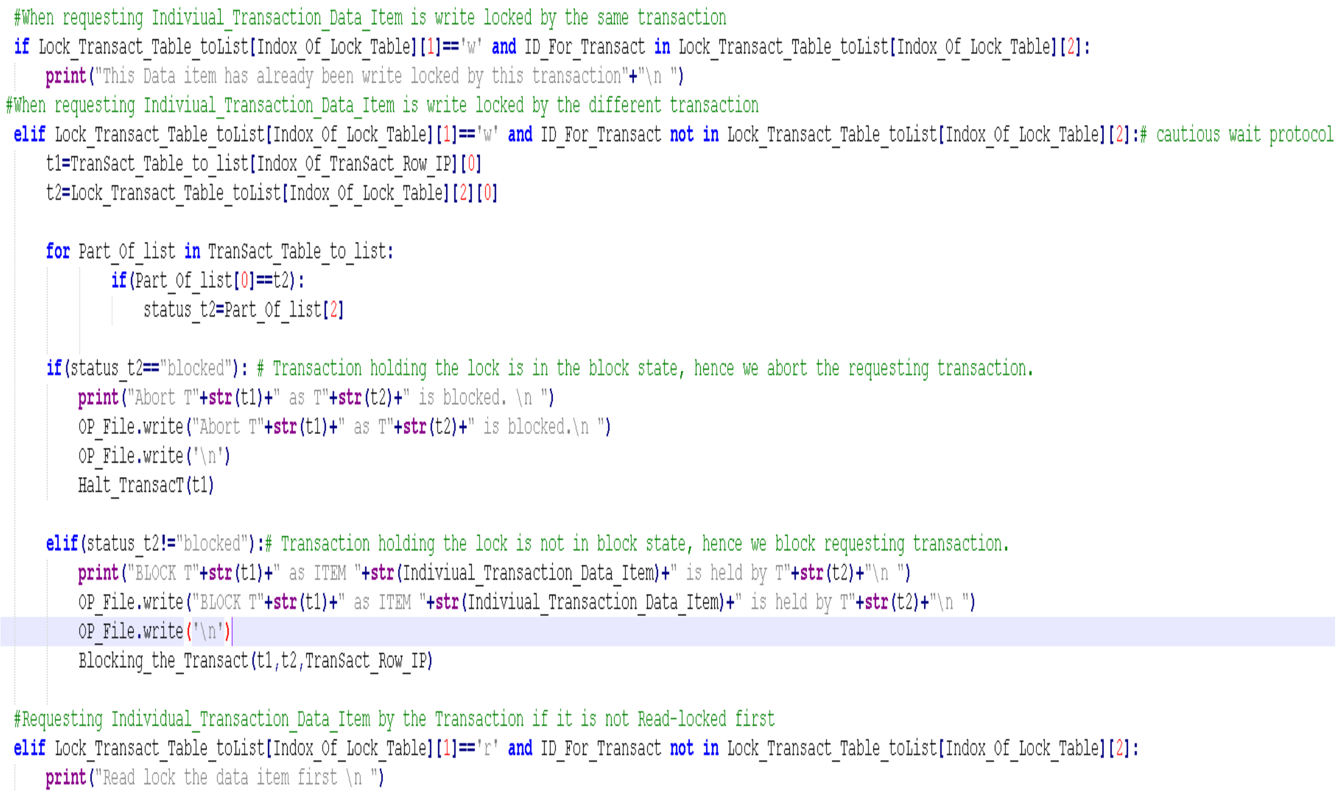
```
def Halt_TransacT(abort_tid):

    global TranSact_Table_to_list
    global Lock_Transact_Table_toList
    OP_Blocked=[]

    for q in range(len(TranSact_Table_to_list)):

        if TranSact_Table_to_list[q][0]==abort_tid:

            if(TranSact_Table_to_list[q][2]!='committed' and TranSact_Table_to_list[q][2]!='aborted'):

                #updating the TranSact_Table_to_list

                TranSact_Table_to_list[q][2]='aborted'
                TranSact_Table_to_list[q][3]=[]
                TranSact_Table_to_list[q][4]=[]

                #updating the Lock_Transact_Table_toList

                for r in range(len(Lock_Transact_Table_toList)):
                    if abort_tid in Lock_Transact_Table_toList[r][2]:
                        if len(Lock_Transact_Table_toList[r][2])==1:
                            Lock_Transact_Table_toList.remove(Lock_Transact_Table_toList[r])
                        else:
                            Lock_Transact_Table_toList[r][2].remove(abort_tid)

    for t in range(len(TranSact_Table_to_list)):
        for x in TranSact_Table_to_list[t][3]:
            if abort_tid==x :

                TranSact_Table_to_list[t][3].remove(abort_tid)

            #Examining whether the aborting transaction has blocked any transactions
                if(TranSact_Table_to_list[t][2]=='Blocked' and len(TranSact_Table_to_list[t][3])==0):

                # The next blocked transaction will be executed and its status will be 'Active'
                    TranSact_Table_to_list[t][2]='active'
                    TranSact_Table_to_list[t][3] = []
                for O_P in TranSact_Table_to_list[t][4]:
                    OP_Blocked.append(O_P)
                TranSact_Table_to_list[t][4]=[]

    for O_P in OP_Blocked:
        Transact_EXecute(O_P)
```

(2) Block Function



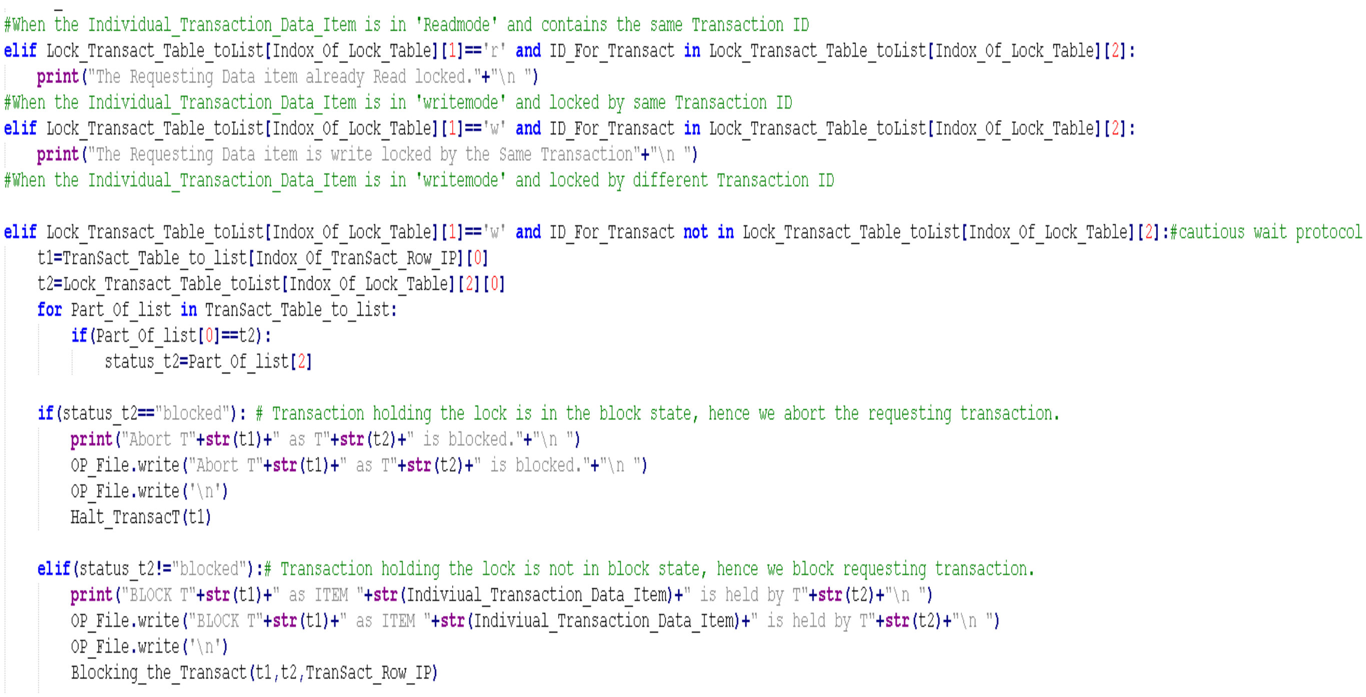
```
def Blocking_the_Transact(blocked_tid,blocked_by_tid,blocked_operation):

    global TranSact_Table_to_list
    blocked_by_tid_list=[]

    # blocked_tid - The transaction id of the transaction that should be blocked
    # blocked_by_tid - The transaction id of the transaction causing the block
    # blocked_operation - Operation that should be blocked

    # Keeping the transaction table updated
    # Reading the items from the transaction table

    for p in range(len(TranSact_Table_to_list)):
        if TranSact_Table_to_list[p][0] == blocked_tid:
            if TranSact_Table_to_list[p][2] == 'active':
                # Changing the status of the transaction table to 'blocked'
                TranSact_Table_to_list[p][2] = 'Blocked'

                # 'blocked by' list for that transaction is updated
                TranSact_Table_to_list[p][3].append(blocked_by_tid)

                # The operation is stored in the 'Blocked Operations' list
                TranSact_Table_to_list[p][4].append(blocked_operation)

            if blocked_by_tid not in TranSact_Table_to_list[p][3]:
                TranSact_Table_to_list[p][3].append(blocked_by_tid)
```

(3) write function



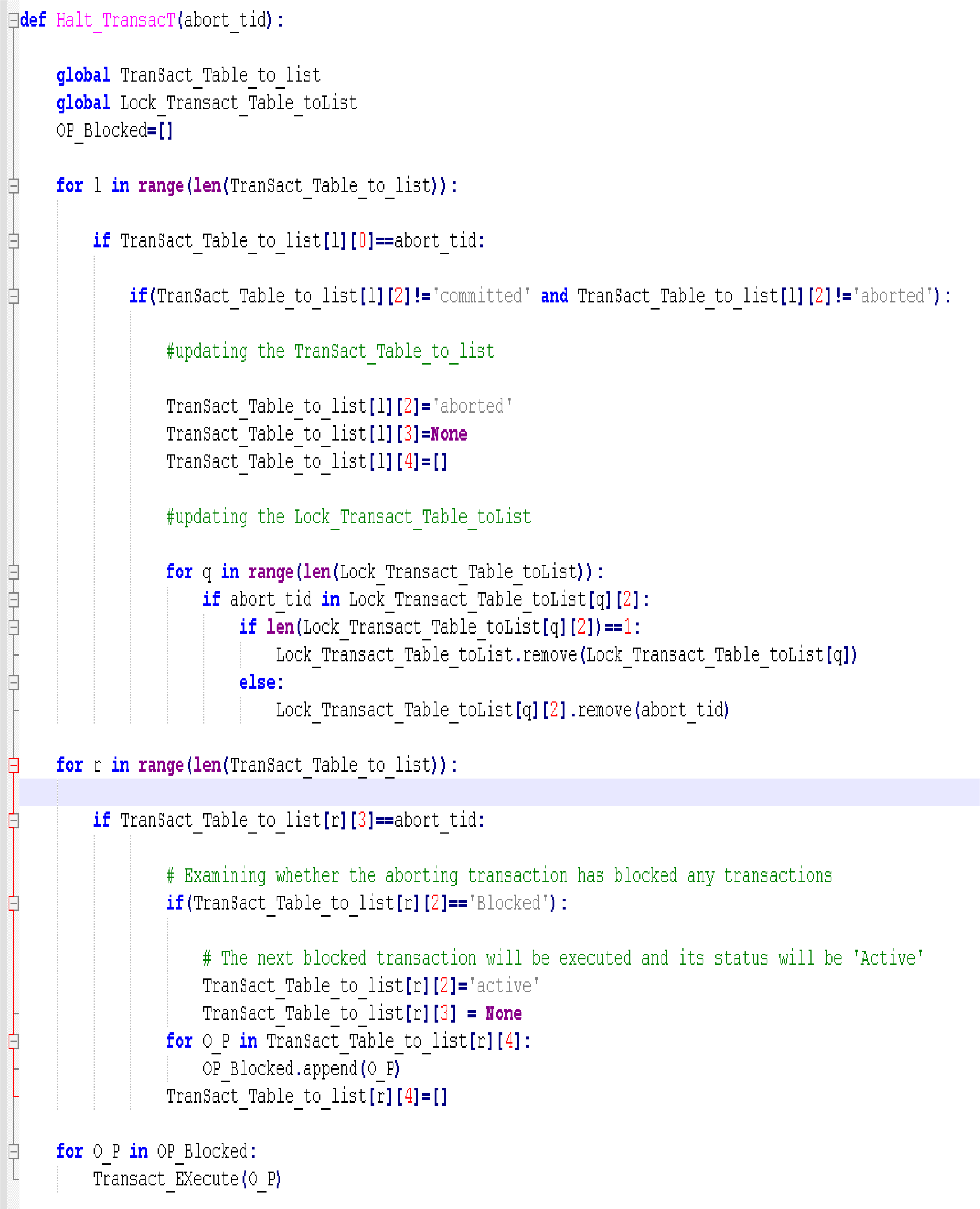
```
#When requesting Indiviual_Transaction_Data_Item is write locked by the same transaction
if Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]=='w' and ID_For_Transact in Lock_Transact_Table_toList[Indox_Of_Lock_Table][2]:
    print("This Data item has already been write locked by this transaction"+"\n ")
#When requesting Indiviual_Transaction_Data_Item is write locked by the different transaction
elif Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]=='w' and ID_For_Transact not in Lock_Transact_Table_toList[Indox_Of_Lock_Table][2]:# cautious wait protocol
    t1=TranSact_Table_to_list[Indox_Of_TranSact_Row_IP][0]
    t2=Lock_Transact_Table_toList[Indox_Of_Lock_Table][2][0]

    for Part_Of_list in TranSact_Table_to_list:
        if(Part_Of_list[0]==t2):
            status_t2=Part_Of_list[2]

    if(status_t2=="blocked"): # Transaction holding the lock is in the block state, hence we abort the requesting transaction.
        print("Abort T"+str(t1)+" as T"+str(t2)+" is blocked. \n ")
        OP_File.write("Abort T"+str(t1)+" as T"+str(t2)+" is blocked.\n ")
        OP_File.write('\n')
        Halt_TransacT(t1)

    elif(status_t2!="blocked"):# Transaction holding the lock is not in block state, hence we block requesting transaction.
        print("BLOCK T"+str(t1)+" as ITEM "+str(Indiviual_Transaction_Data_Item)+" is held by T"+str(t2)+"\n ")
        OP_File.write("BLOCK T"+str(t1)+" as ITEM "+str(Indiviual_Transaction_Data_Item)+" is held by T"+str(t2)+"\n ")
        OP_File.write('\n')
        Blocking_the_Transact(t1,t2,TranSact_Row_IP)

#Requesting Individual_Transaction_Data_Item by the Transaction if it is not Read-locked first
elif Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]=='r' and ID_For_Transact not in Lock_Transact_Table_toList[Indox_Of_Lock_Table][2]:
    print("Read lock the data item first \n ")
```

(4) Read Funtcion



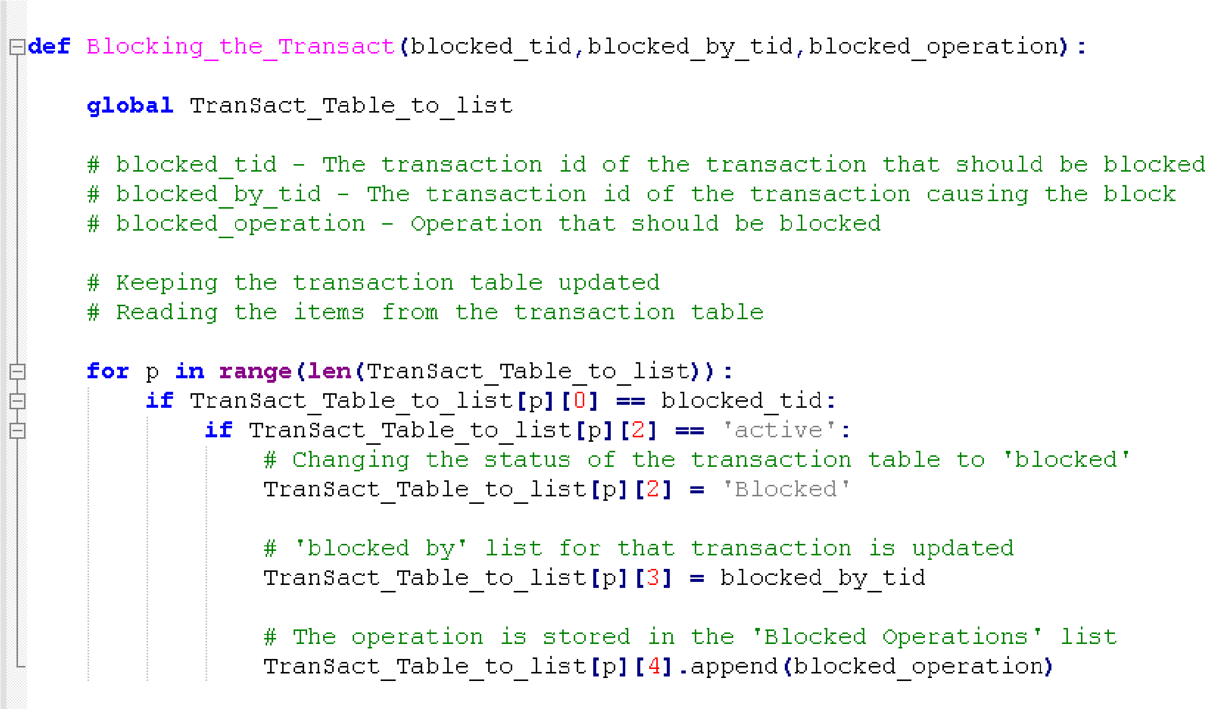
```
#When the Individual_Transaction_Data_Item is in 'Readmode' and contains the same Transaction ID
elif Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]=='r' and ID_For_Transact in Lock_Transact_Table_toList[Indox_Of_Lock_Table][2]:
    print("The Requesting Data item already Read locked."+"\n ")
#When the Individual_Transaction_Data_Item is in 'writemode' and locked by same Transaction ID
elif Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]=='w' and ID_For_Transact in Lock_Transact_Table_toList[Indox_Of_Lock_Table][2]:
    print("The Requesting Data item is write locked by the Same Transaction"+"\n ")
#When the Individual_Transaction_Data_Item is in 'writemode' and locked by different Transaction ID

elif Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]=='w' and ID_For_Transact not in Lock_Transact_Table_toList[Indox_Of_Lock_Table][2]:#cautious wait protocol
    t1=TranSact_Table_to_list[Indox_Of_TranSact_Row_IP][0]
    t2=Lock_Transact_Table_toList[Indox_Of_Lock_Table][2][0]
    for Part_Of_list in TranSact_Table_to_list:
        if(Part_Of_list[0]==t2):
            status_t2=Part_Of_list[2]

    if(status_t2=="blocked"): # Transaction holding the lock is in the block state, hence we abort the requesting transaction.
        print("Abort T"+str(t1)+" as T"+str(t2)+" is blocked."+"\n ")
        OP_File.write("Abort T"+str(t1)+" as T"+str(t2)+" is blocked."+"\n ")
        OP_File.write('\n')
        Halt_TransacT(t1)

    elif(status_t2!="blocked"):# Transaction holding the lock is not in block state, hence we block requesting transaction.
        print("BLOCK T"+str(t1)+" as ITEM "+str(Indiviual_Transaction_Data_Item)+" is held by T"+str(t2)+"\n ")
        OP_File.write("BLOCK T"+str(t1)+" as ITEM "+str(Indiviual_Transaction_Data_Item)+" is held by T"+str(t2)+"\n ")
        OP_File.write('\n')
        Blocking_the_Transact(t1,t2,TranSact_Row_IP)
```

CODE FOR WAIT-DIE PROTOCOL: (1) Abort Function



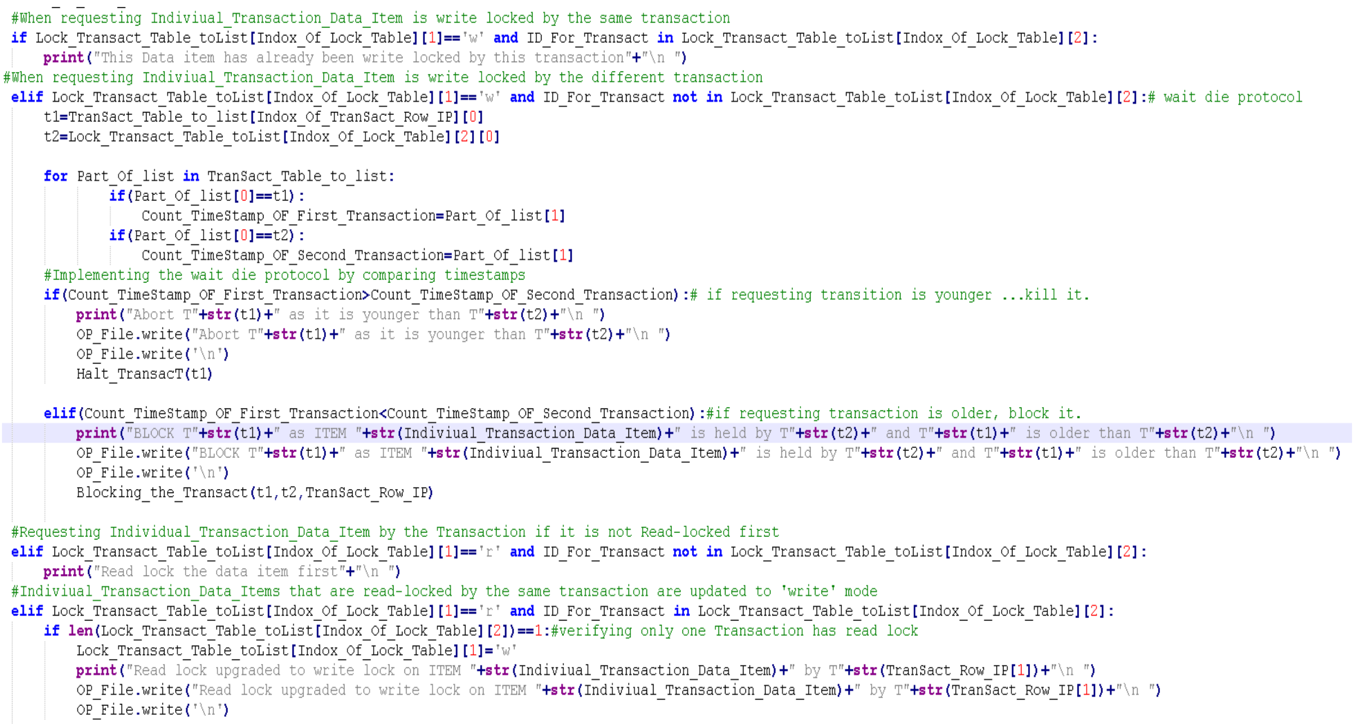
```
def Halt_TransacT(abort_tid):

    global TranSact_Table_to_list
    global Lock_Transact_Table_toList
    OP_Blocked=[]

    for l in range(len(TranSact_Table_to_list)):

        if TranSact_Table_to_list[l][0]==abort_tid:

            if(TranSact_Table_to_list[l][2]!='committed' and TranSact_Table_to_list[l][2]!='aborted'):

                #updating the TranSact_Table_to_list

                TranSact_Table_to_list[l][2]='aborted'
                TranSact_Table_to_list[l][3]=None
                TranSact_Table_to_list[l][4]=[]

                #updating the Lock_Transact_Table_toList

                for q in range(len(Lock_Transact_Table_toList)):
                    if abort_tid in Lock_Transact_Table_toList[q][2]:
                        if len(Lock_Transact_Table_toList[q][2])==1:
                            Lock_Transact_Table_toList.remove(Lock_Transact_Table_toList[q])
                        else:
                            Lock_Transact_Table_toList[q][2].remove(abort_tid)

    for r in range(len(TranSact_Table_to_list)):

        if TranSact_Table_to_list[r][3]==abort_tid:

                # Examining whether the aborting transaction has blocked any transactions
                if(TranSact_Table_to_list[r][2]=='Blocked'):

                    # The next blocked transaction will be executed and its status will be 'Active'
                    TranSact_Table_to_list[r][2]='active'
                    TranSact_Table_to_list[r][3] = None
                for O_P in TranSact_Table_to_list[r][4]:
                    OP_Blocked.append(O_P)
                TranSact_Table_to_list[r][4]=[]

    for O_P in OP_Blocked:
        Transact_EXecute(O_P)
```

(2) Block Function



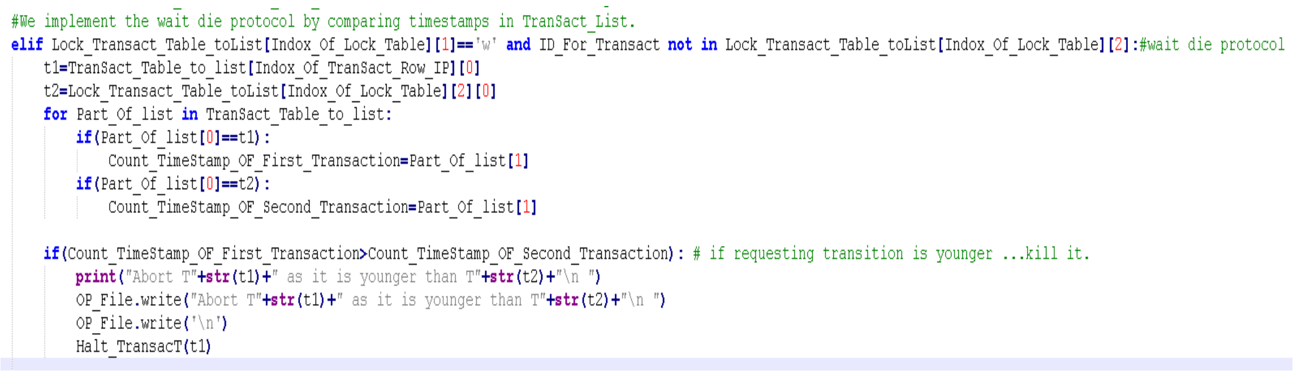
```
def Blocking_the_Transact(blocked_tid,blocked_by_tid,blocked_operation):

    global TranSact_Table_to_list

    # blocked_tid - The transaction id of the transaction that should be blocked
    # blocked_by_tid - The transaction id of the transaction causing the block
    # blocked_operation - Operation that should be blocked

    # Keeping the transaction table updated
    # Reading the items from the transaction table

    for p in range(len(TranSact_Table_to_list)):
        if TranSact_Table_to_list[p][0] == blocked_tid:
            if TranSact_Table_to_list[p][2] == 'active':
                # Changing the status of the transaction table to 'blocked'
                TranSact_Table_to_list[p][2] = 'Blocked'

                # 'blocked by' list for that transaction is updated
                TranSact_Table_to_list[p][3] = blocked_by_tid

                # The operation is stored in the 'Blocked Operations' list
                TranSact_Table_to_list[p][4].append(blocked_operation)
```

(3) write function



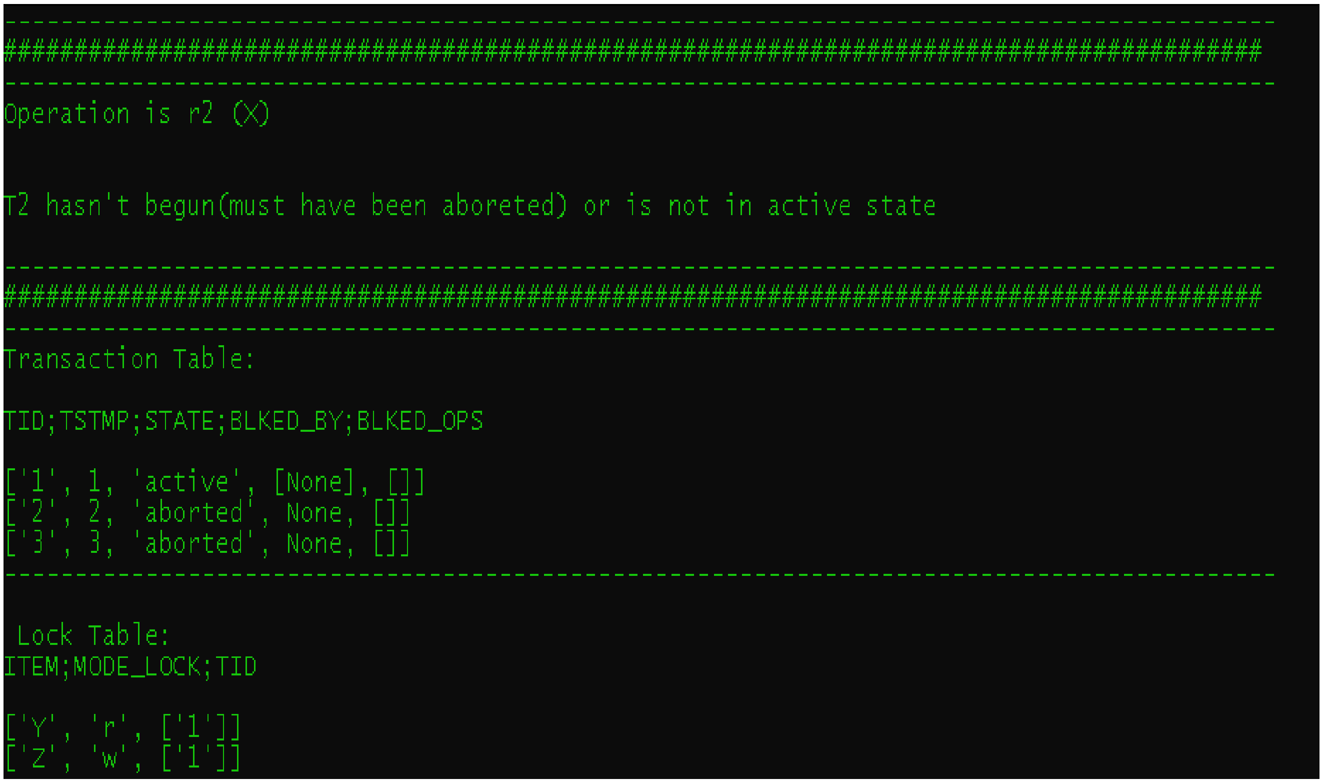
```
 #When requesting Indiviual_Transaction_Data_Item is write locked by the same transaction
 if Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]=='w' and ID_For_Transact in Lock_Transact_Table_toList[Indox_Of_Lock_Table][2]:
     print("This Data item has already been write locked by this transaction"+"\n ")
#When requesting Indiviual_Transaction_Data_Item is write locked by the different transaction
 elif Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]=='w' and ID_For_Transact not in Lock_Transact_Table_toList[Indox_Of_Lock_Table][2]:# wait die protocol
     t1=TranSact_Table_to_list[Indox_Of_TranSact_Row_IP][0]
     t2=Lock_Transact_Table_toList[Indox_Of_Lock_Table][2][0]

     for Part_Of_list in TranSact_Table_to_list:
             if(Part_Of_list[0]==t1):
                 Count_TimeStamp_OF_First_Transaction=Part_Of_list[1]
             if(Part_Of_list[0]==t2):
                 Count_TimeStamp_OF_Second_Transaction=Part_Of_list[1]
     #Implementing the wait die protocol by comparing timestamps
     if(Count_TimeStamp_OF_First_Transaction>Count_TimeStamp_OF_Second_Transaction):# if requesting transition is younger ...kill it.
         print("Abort T"+str(t1)+" as it is younger than T"+str(t2)+"\n ")
         OP_File.write("Abort T"+str(t1)+" as it is younger than T"+str(t2)+"\n ")
         OP_File.write('\n')
         Halt_TransacT(t1)

     elif(Count_TimeStamp_OF_First_Transaction<Count_TimeStamp_OF_Second_Transaction):#if requesting transaction is older, block it.
         print("BLOCK T"+str(t1)+" as ITEM "+str(Indiviual_Transaction_Data_Item)+" is held by T"+str(t2)+" and T"+str(t1)+" is older than T"+str(t2)+"\n ")
         OP_File.write("BLOCK T"+str(t1)+" as ITEM "+str(Indiviual_Transaction_Data_Item)+" is held by T"+str(t2)+" and T"+str(t1)+" is older than T"+str(t2)+"\n ")
         OP_File.write('\n')
         Blocking_the_Transact(t1,t2,TranSact_Row_IP)

 #Requesting Individual_Transaction_Data_Item by the Transaction if it is not Read-locked first
 elif Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]=='r' and ID_For_Transact not in Lock_Transact_Table_toList[Indox_Of_Lock_Table][2]:
     print("Read lock the data item first"+"\n ")
 #Indiviual_Transaction_Data_Items that are read-locked by the same transaction are updated to 'write' mode
 elif Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]=='r' and ID_For_Transact in Lock_Transact_Table_toList[Indox_Of_Lock_Table][2]:
     if len(Lock_Transact_Table_toList[Indox_Of_Lock_Table][2])==1:#verifying only one Transaction has read lock
         Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]='w'
         print("Read lock upgraded to write lock on ITEM "+str(Indiviual_Transaction_Data_Item)+" by T"+str(TranSact_Row_IP[1])+"\n ")
         OP_File.write("Read lock upgraded to write lock on ITEM "+str(Indiviual_Transaction_Data_Item)+" by T"+str(TranSact_Row_IP[1])+"\n ")
         OP_File.write('\n')
```

(4) Read Funtcion



```
#We implement the wait die protocol by comparing timestamps in TranSact_List.
elif Lock_Transact_Table_toList[Indox_Of_Lock_Table][1]=='w' and ID_For_Transact not in Lock_Transact_Table_toList[Indox_Of_Lock_Table][2]:#wait die protocol
    t1=TranSact_Table_to_list[Indox_Of_TranSact_Row_IP][0]
    t2=Lock_Transact_Table_toList[Indox_Of_Lock_Table][2][0]
    for Part_Of_list in TranSact_Table_to_list:
        if(Part_Of_list[0]==t1):
            Count_TimeStamp_OF_First_Transaction=Part_Of_list[1]
        if(Part_Of_list[0]==t2):
            Count_TimeStamp_OF_Second_Transaction=Part_Of_list[1]

    if(Count_TimeStamp_OF_First_Transaction>Count_TimeStamp_OF_Second_Transaction): # if requesting transition is younger ...kill it.
        print("Abort T"+str(t1)+" as it is younger than T"+str(t2)+"\n ")
        OP_File.write("Abort T"+str(t1)+" as it is younger than T"+str(t2)+"\n ")
        OP_File.write('\n')
        Halt_TransacT(t1)
```

TRANSACTION TABLE AND LOCK TABLE OUTPUT (SAMPLE):



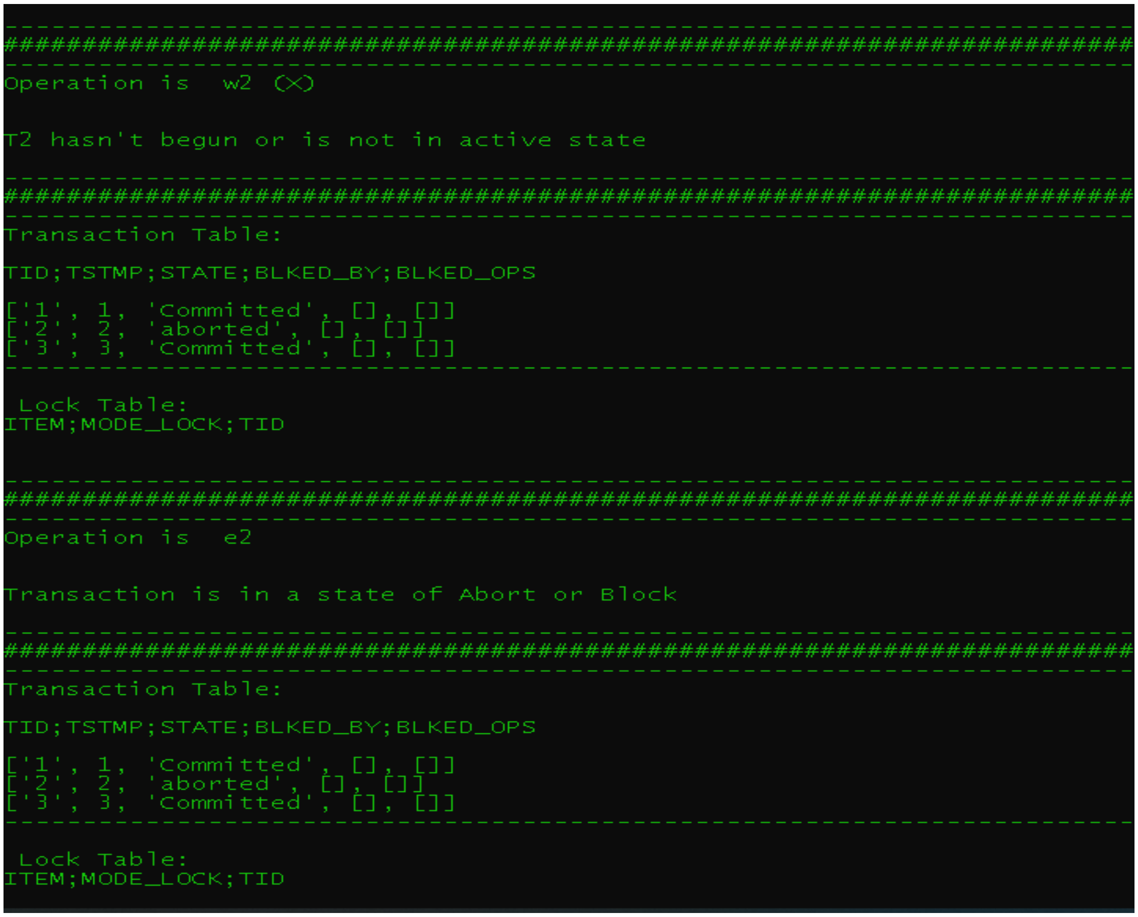
```
--------------------------------------------------------------------------------
################################################################################
--------------------------------------------------------------------------------
Operation is r2 (X)


T2 hasn't begun(must have been aboreted) or is not in active state

--------------------------------------------------------------------------------
################################################################################
--------------------------------------------------------------------------------
Transaction Table:

TID;TSTMP;STATE;BLKED_BY;BLKED_OPS

['1', 1, 'active', [None], []]
['2', 2, 'aborted', None, []]
['3', 3, 'aborted', None, []]
--------------------------------------------------------------------------------

 Lock Table:
ITEM;MODE_LOCK;TID

['Y', 'r', ['1']]
['Z', 'w', ['1']]
```

Test Cases for Sample Inputs(Cautious-wait):

- **Input 1:**

  b1;
  r1 (Y);
  r1 (Z);
  b2;
  r2 (Y);
  b3;
  r3 (Y);
  w1 (Z);
  w3 (Y);
  w2 (Y);
  r2 (X);
  e1;
  e3;
  w2 (X);
  e2;

  Output:(displaying only final lines of output as output is very large)

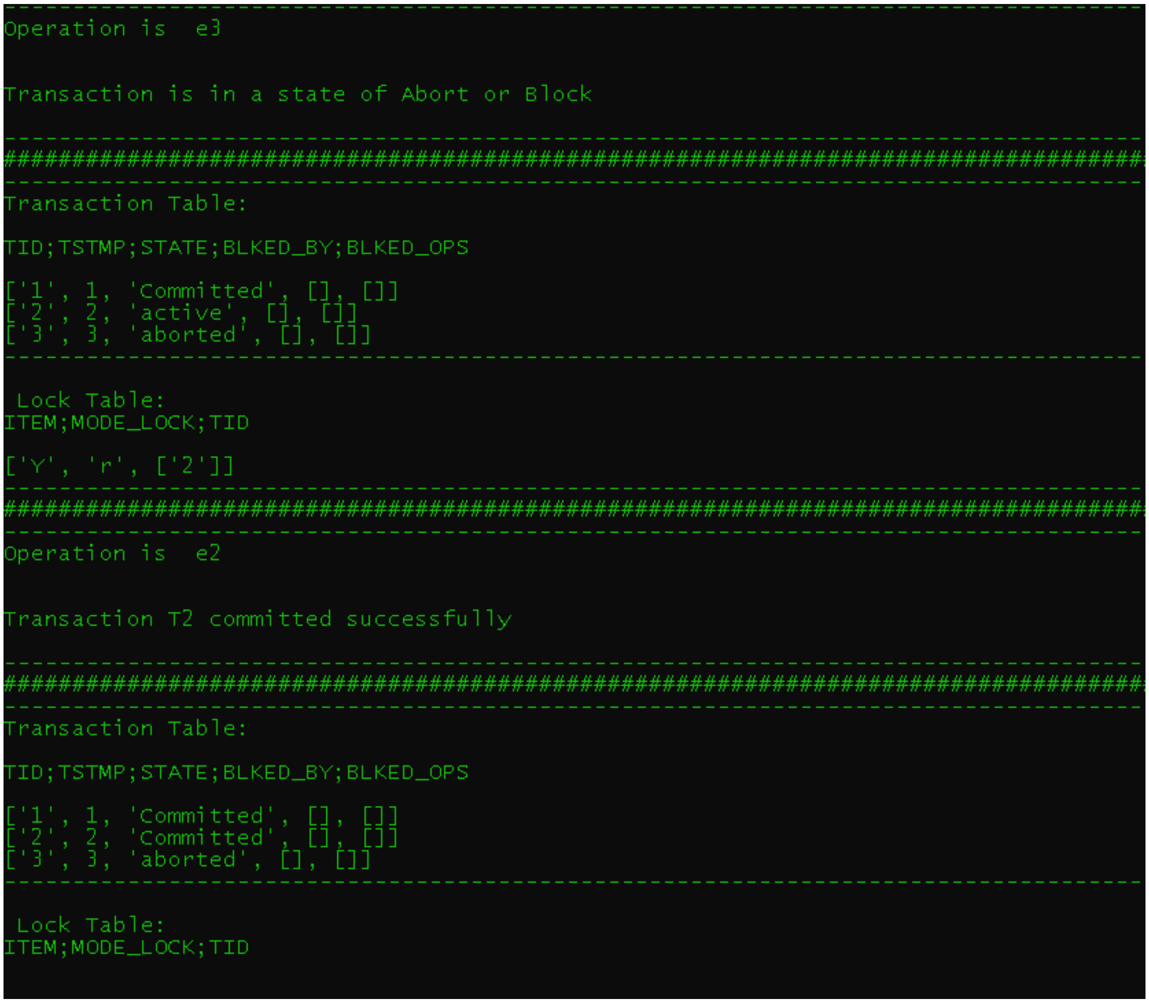


```
--------------------------------------------------------------------------------
################################################################################
--------------------------------------------------------------------------------
Operation is  w2 (X)

T2 hasn't begun or is not in active state

--------------------------------------------------------------------------------
################################################################################
--------------------------------------------------------------------------------
Transaction Table:

TID;TSTMP;STATE;BLKED_BY;BLKED_OPS

['1', 1, 'Committed', [], []]
['2', 2, 'aborted', [], []]
['3', 3, 'Committed', [], []]
--------------------------------------------------------------------------------

 Lock Table:
ITEM;MODE_LOCK;TID


--------------------------------------------------------------------------------
################################################################################
--------------------------------------------------------------------------------
Operation is  e2

Transaction is in a state of Abort or Block

--------------------------------------------------------------------------------
################################################################################
--------------------------------------------------------------------------------
Transaction Table:

TID;TSTMP;STATE;BLKED_BY;BLKED_OPS

['1', 1, 'Committed', [], []]
['2', 2, 'aborted', [], []]
['3', 3, 'Committed', [], []]
--------------------------------------------------------------------------------

 Lock Table:
ITEM;MODE_LOCK;TID
```

- **Input 2:**

b1;
r1(Y);
w1(Y);
r1(Z);
b2;
r2(Y);
b3;
r3(Z);
w1(Z);
e1;
w3(Z);
e3;
e2;

Output:(displaying only final lines of output as output is very large)



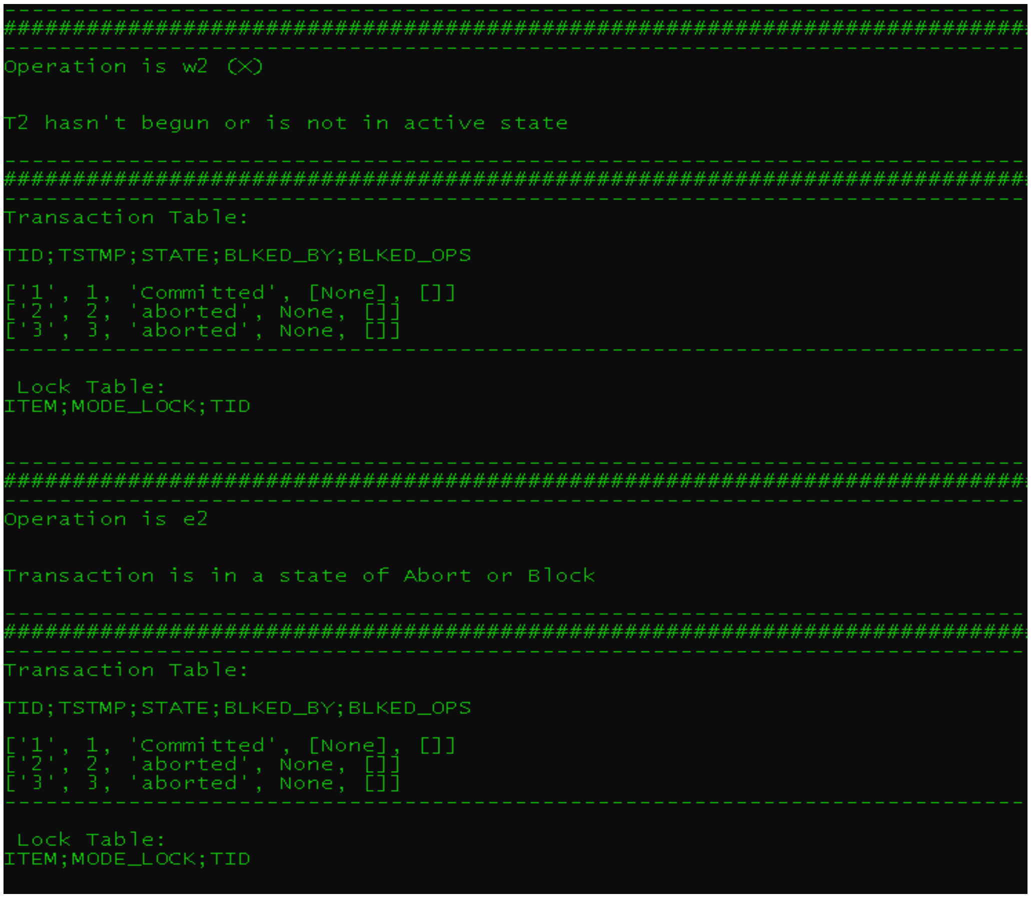
```
-----------------------------------------------------------------------------------
Operation is  e3


Transaction is in a state of Abort or Block

-----------------------------------------------------------------------------------
###################################################################################
-----------------------------------------------------------------------------------
Transaction Table:

TID;TSTMP;STATE;BLKED_BY;BLKED_OPS

['1', 1, 'Committed', [], []]
['2', 2, 'active', [], []]
['3', 3, 'aborted', [], []]
-----------------------------------------------------------------------------------

 Lock Table:
ITEM;MODE_LOCK;TID

['Y', 'r', ['2']]
-----------------------------------------------------------------------------------
###################################################################################
-----------------------------------------------------------------------------------
Operation is  e2


Transaction T2 committed successfully

-----------------------------------------------------------------------------------
###################################################################################
-----------------------------------------------------------------------------------
Transaction Table:

TID;TSTMP;STATE;BLKED_BY;BLKED_OPS

['1', 1, 'Committed', [], []]
['2', 2, 'Committed', [], []]
['3', 3, 'aborted', [], []]
-----------------------------------------------------------------------------------

 Lock Table:
ITEM;MODE_LOCK;TID
```

Test Cases for Sample Inputs(Wait-Die):

- **Input 1:**

  b1;
  r1 (Y);
  r1 (Z);
  b2;
  r2 (Y);
  b3;
  r3 (Y);
  w1 (Z);
  w3 (Y);
  w2 (Y);
  r2 (X);
  e1;
  e3;
  w2 (X);
  e2;

  Output:(displaying only final lines of output as output is very large)

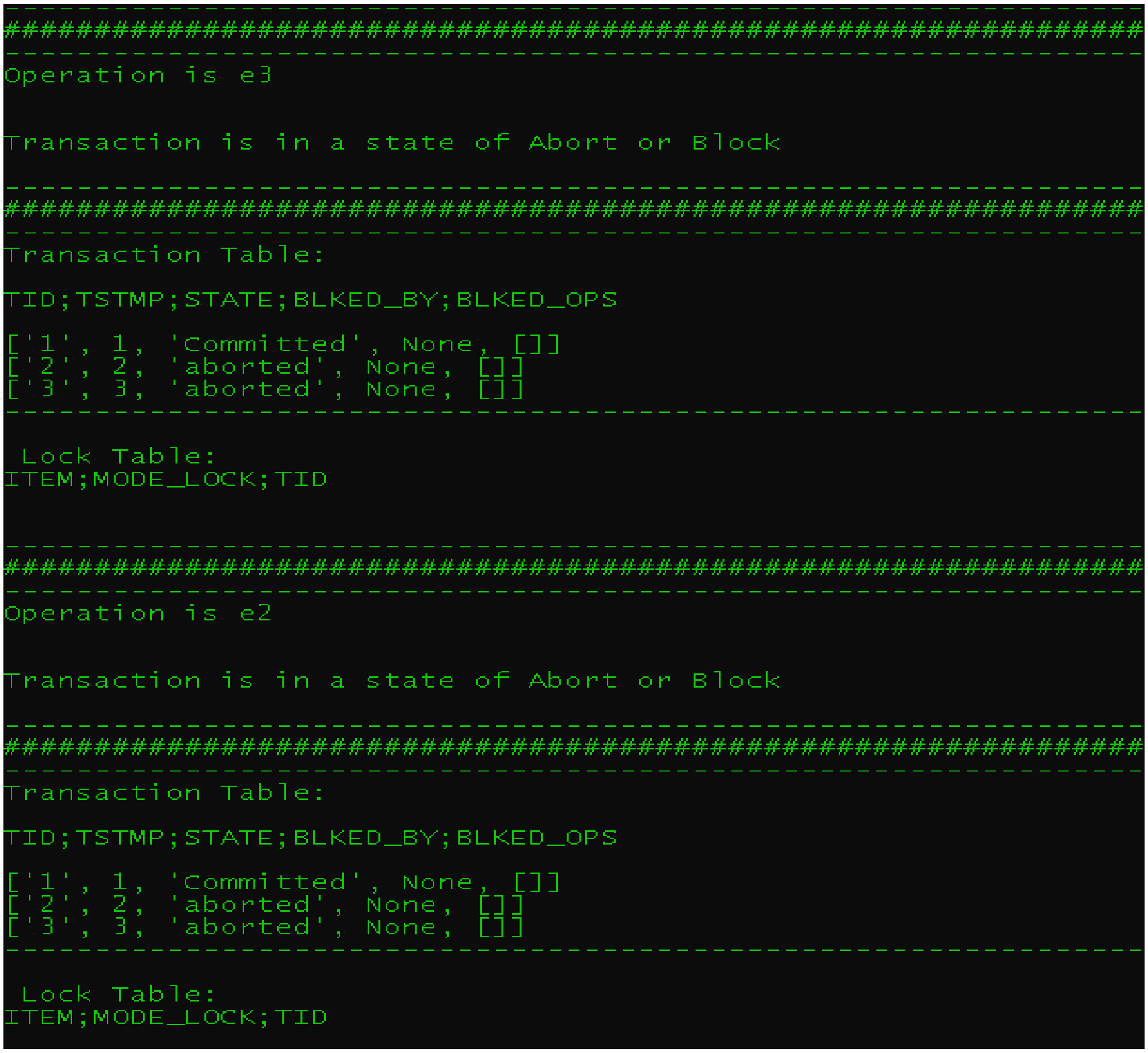


```
---------------------------------------------------------------------------
###########################################################################
---------------------------------------------------------------------------
Operation is w2 (X)

T2 hasn't begun or is not in active state

---------------------------------------------------------------------------
###########################################################################
---------------------------------------------------------------------------
Transaction Table:

TID;TSTMP;STATE;BLKED_BY;BLKED_OPS

['1', 1, 'Committed', [None], []]
['2', 2, 'aborted', None, []]
['3', 3, 'aborted', None, []]
---------------------------------------------------------------------------

 Lock Table:
ITEM;MODE_LOCK;TID


---------------------------------------------------------------------------
###########################################################################
---------------------------------------------------------------------------
Operation is e2

Transaction is in a state of Abort or Block

---------------------------------------------------------------------------
###########################################################################
---------------------------------------------------------------------------
Transaction Table:

TID;TSTMP;STATE;BLKED_BY;BLKED_OPS

['1', 1, 'Committed', [None], []]
['2', 2, 'aborted', None, []]
['3', 3, 'aborted', None, []]
---------------------------------------------------------------------------

 Lock Table:
ITEM;MODE_LOCK;TID
```

- **Input 2:**

b1;
r1(Y);
w1(Y);
r1(Z);
b2;
r2(Y);
b3;
r3(Z);
w1(Z);
e1;
w3(Z);
e3;
e2;

Output:(displaying only final lines of output as output is very large)



```
---------------------------------------------------------------
###############################################################
---------------------------------------------------------------
Operation is e3

Transaction is in a state of Abort or Block

---------------------------------------------------------------
###############################################################
---------------------------------------------------------------
Transaction Table:

TID;TSTMP;STATE;BLKED_BY;BLKED_OPS

['1', 1, 'Committed', None, []]
['2', 2, 'aborted', None, []]
['3', 3, 'aborted', None, []]
---------------------------------------------------------------

 Lock Table:
ITEM;MODE_LOCK;TID


---------------------------------------------------------------
###############################################################
---------------------------------------------------------------
Operation is e2

Transaction is in a state of Abort or Block

---------------------------------------------------------------
###############################################################
---------------------------------------------------------------
Transaction Table:

TID;TSTMP;STATE;BLKED_BY;BLKED_OPS

['1', 1, 'Committed', None, []]
['2', 2, 'aborted', None, []]
['3', 3, 'aborted', None, []]
---------------------------------------------------------------

 Lock Table:
ITEM;MODE_LOCK;TID
```