BASIC TIMESTAMP ORDERING PROTOCOL

The main idea for this protocol is to order the transactions based on their Timestamps. A schedule in which the transactions participate is then serialisable and the only equivalent serial schedule permitted has the transactions in the order of their Timestamp Values. Stating simply, the schedule is equivalent to the particular Serial Order corresponding to the order of the Transaction timestamps.
Whenever the Basic TO algorithm detects two conflicting operations occur in incorrect order, it rejects the later of the two operation by aborting the Transaction that issued it. Schedules produced by Basic TO are guaranteed to be conflict serialisable. Using Timestamp, one can ensure that our schedule will be deadlock free.

CODE

The code is written in python. Based on the input given by the user “ python bto.py -x 2 -i 2 -v 1“ , a transaction table will be created (List of list OP) .

-x : Number of transactions
-i : Max number of instructions .
-v : Number of variables
-t : Total time of running (default 5 sec)

The list of list OP contains all the operations in the order of their arrival. Each entry of OP is in the following way

< Tx Name | Operation (R/W) | Variable name | Write value incase of write| timestamp of instruction | status (1 for active , -1 for aborted / invalidated transaction operation) >

The code iterates through all the instructions, performs the respective operations and checks for abort condition. In case of abort, all the operations of that transaction (say T1) are invalidated and the value of all variables are restored to the original state . The execution is again started from the timestamp of T1.

SOME SPECIFIC EXAMPLES OF SCHEDULES

eg 1:-
OP=[[1, 'W', 0, 6, 0, 1],[0, 'W', 0, 4, 1, 1],[1, 'R', 0, 2, 1],[0, 'W', 0, 3, 3, 1],[1, 'W', 0, 7, 4, 1]]

T0	**T1**	
	W(x)=6	t=0
W(x)=4		t=1
	R(x)	t=2
W(x)=3		t=3
	W(x)=7	t=4

The output is as follows: - (Consider x to be the variable 0)

```
i=0
Writing value of variable 0 as 6 by T1

i=1
 Writing value of variable 0 as 4 by T0

i=2
aborting.. Transaction T1 and restoring

i=0
i=1
 Writing value of variable 0 as 4 by T0

i=2
i=3
 Writing value of variable 0 as 3 by T0

i=4
[3]
```

So the final output is the write performed by T0 = 3

Eg-2

OP =[[1, 'W', 1, 5, 0, 1], [1, 'R', 1, 1, 1], [1, 'R', 0, 2, 1], [0, 'W', 0, 4, 3, 1], [1, 'R', 1, 4, 1]]

T0	**T1**	
	W(y)=5	t=0
	R(y)	t=1
	R(x)	t=2
W(x)=4		t=3
	R(y)	t=4

Output:

```
i=0
 Writing value of variable 1 as 5 by T1

i=1
 Reading value of variable 1 as 5 by T1

i=2
 Reading value of variable 0 as -1 by T1

i=3
 Writing value of variable 0 as 4 by T0

i=4
 Reading value of variable 1 as 5 by T1

[4, 5]
```

Here 4 is the value taken by variable 0 (x) and 5 by variable 1 (y).

The second example is conflict serialisable and hence no transaction aborts.

Group 4 :

Pulkit Sapra (B15123)
Jonty Purbia
Avnish Kumar
Pramod Jonwal