Name: SOLUTIONS ____ netid:___________________

CS 425/ECE 428 Distributed Systems
Mid-Term Exam 2

Total points: 50

Duration: 75 minutes

The exam includes 9 questions.

1. Consider a Chord peer-to-peer network consisting of 5 nodes that use 6-bit identifiers. The node identifiers are N12, N15, N29, N33 and N54.

   (a) (3 points) Show the finger table at node N15 using the format below.

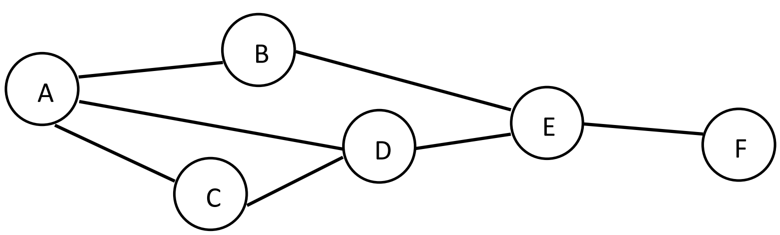
i	ft[i]
0	29
1	29
2	29
3	29
4	29
5	54

   (b) (2 points) If node N15 wants to locate key K6, identify the node to which N15 will send its request message.

   Node N54

2. (5 points) Assume the following: (i) node D below can sign its messages using its private key, (ii) the network contains at most 1 Byzantine faulty node, and (iii) all nodes know D's public key.

   Answer True or False in each part below.



(a) Node B can always reliably receive messages sent by node D. TRUE

(b) Node F can always reliably receive messages sent by node D. FALSE

3. (a) (3 points) In the figure below, identify conflicting operations of transactions T and U.

   (b) (2 points) Is the interleaving of transactions T and U below *serially equivalent*?
   Answer YES or NO ____YES__________

Transaction T	**Transaction U**
openTransaction	
	openTransaction
	write(x,1)
a = read(x)	
	write(z,6)
write(y,2)	
	write(w,5)
	b = read(x)
c = read(w)	
	closeTransaction
closeTransaction	

4. (6 points) Does the shared memory algorithm presented below guarantee that a process that wants to enter the critical section will eventually enter the critical section? Assume that flag is initialize to 0. Answer Yes or No, and **justify your answer**.

Code for entry section:

1 Wait until flag = 0
2 flag = 1

Code for exit section:

3 flag = flag - 1

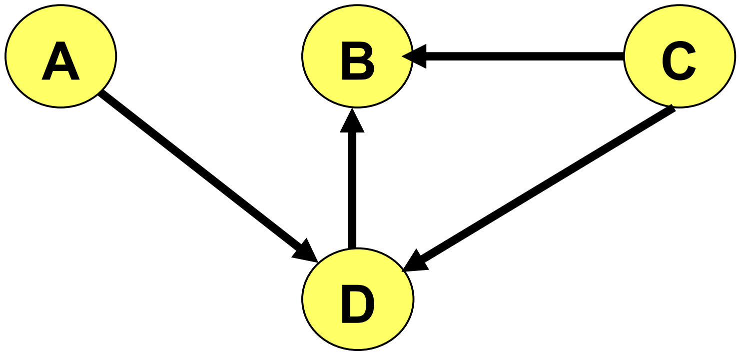
NO.

Two processes may both find flag = 0 on line 1, and both proceed to line 2, and then both set flag = 1, and then both enter critical section.

When they both exit, the decrement flag by 1 in line 3, and the fina value of flag is now -1.

All processs wanting to enter the critical section subsequently will wait at line 1 indefinitely, since flag is now -1.

5. (5 points) The network below uses the link reversal algorithm to maintain routes to **destination B**. The figure shows the directions assigned to the links at a particular time.



Show how the link reversal algorithm will change directions of the links, if link BD now breaks. Present your answer in the form of a sequence of network states.

In the first step D reverses all its links.

After that, A reverses its link.

6. (6 points) Consider the Bakery algorithm for mutual exclusion (included in the attached handout).

   Suppose that the process with the largest identifier is *Byzantine faulty*, and may write arbitrary values to the shared memory.

   In this system, is it possible for the faulty process to prevent non-faulty processes from entering the critical section?

   Justify your answer.

   There are several possible answer.

   One possibility is for a faulty process i to sets choosing[i] flag to true forever.

7. (6 points) This question relates to the Paxos algorithm discussed in class.

State True or False:

(a) Paxos is guaranteed to perform correctly even if some of the acceptors are Byzantine faulty. FALSE

(b) In an execution in which only crash failures occur, it is not possible for different learners to learn different chosen values. TRUE

(c) If an acceptor responds to a *prepare* request with sequence number *n*, then no proposer will later send that acceptor a *prepare* request with sequence number less than *n*. FALSE

8. (6 points) Determine whether the interleaving of operations of transactions T and U shown below can occur in each of the three cases below. In each case, answer YES or NO.

(a) Read-write locks are used with strict two-phase locking. YES

(b) Exclusive locks are used with two-phase locking. NO

(c) Exclusive locks are used with strict two-phase locking. NO

Transaction T	Transaction U
openTransaction	
	openTransaction
	write (y,5)
a = read (x)	
write (w,2)	
	c = read (x)
	closeTransaction
d = read (y)	
closeTransaction	

9. (6 points) The table below shows the interleaving of the operations performed by transactions T, U and V. Suppose that optimistic concurrency control with ***backward*** validation is used.

Transaction T	Transaction U	Transaction V
openTransaction		
	openTransaction	
		openTransaction
write (z,9)		
a = read (v)		
write (y,7)		
	write(v,8)	
	f = read(w)	
	write(w,8)	
		c = read (v)
		write (w,5)
		closeTransaction
	write(p,2)	
	closeTransaction	
write(q,3)		
closeTransaction		

(a) Will transaction U be aborted? Answer YES or NO, and **briefly explain why.**

YES.

Transaction V commits, and read set of U intersects with write set of V.

(b) Will transaction T be aborted? Answer YES or NO, and **briefly explain why.**

NO.

Transaction V commits, and transaction U aborts, as noted above.

Read set of T does not intersect with write set of V.

Handout for Exam 2

Bakery Algorithm

1

Code for entry section:

```
Choosing[i] := true
Number[i] := max{Number[0], …, Number[n-1]} + 1
Choosing[i] := false
for j := 0 to n-1 (except i) do
    wait until Choosing[j] = false
    wait until Number[j] = 0 or
     (Number[j],j) > (Number[i],i)
endfor
```

Code for exit section:

```
Number[i] := 0
```

2-Processor Mutex Algorithm

2

Code for entry section:

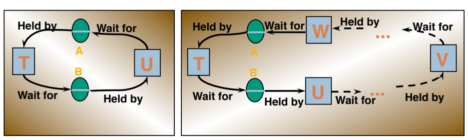
```
1  W[i] := 0
2  wait until W[1-i] = 0 or Priority = i
3  W[i] := 1
4  if (Priority = 1-i) then
5       if (W[1-i] = 1) then goto Line 1
6  else wait until (W[1-i] = 0)
```

Code for exit section:

```
7  Priority := 1-i
8  W[i] := 0
```

Deadlocks

- Necessary conditions for deadlocks
  - Non-shareable resources (locked objects)
  - No preemption on locks
  - Hold & Wait
  - Circular Wait (Wait-for graph)



Validation of Transactions

Backward validation of transaction T_v

boolean valid = true;

for (int T_i = $startTn$+1; T_i <= $finishTn$; T_i++){

if (read set of T_v intersects write set of T_i) valid = false;

}

Forward validation of transaction T_v

boolean valid = true;

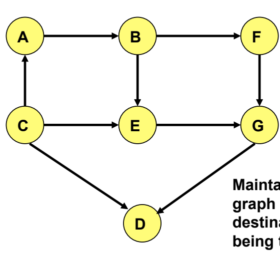
for (int T_{id} = $active1$; T_{id} <= $activeN$; T_{id}++){

if (write set of T_v intersects read set of T_{id}) valid = false;

}

Link Reversal Algorithm

5



Links are bi-directional

But algorithm imposes logical directions on them

Maintain a directed acyclic graph (DAG) for each destination, with the destination being the *only sink*

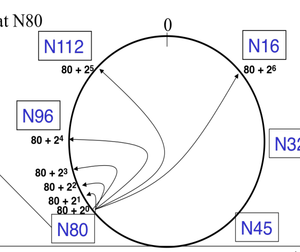
This DAG is for *destination node D*

Peer pointers (2): finger tables

Say m=7

Finger Table at N80

i	$ft[i]$
0	96
1	96
2	96
3	96
4	96
5	112
6	16



ith entry at peer with id n is first peer with id >= $n + 2^i (\mathrm{mod}\, 2^m)$