Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2006

Quiz II

There are 14 questions and 15 pages in this quiz booklet. Answer each question according to the instructions given. You have **50 minutes** to answer the questions.

Most questions are multiple-choice questions. Next to each choice, circle the word **True** or **False**, as appropriate. A correct choice will earn positive points, a wrong choice may earn negative points, and not picking a choice will score 0. The exact number of positive and negative points for each choice in a question depends on the question and choice. The maximum score for each question is given near each question; the minimum for each question is 0. Some questions are harder than others and some questions earn more points than others—you may want to skim all questions before starting.

If you find a question ambiguous, be sure to write down any assumptions you make. **Be neat and legible.** If we can't understand your answer, we can't give you credit!

Write your name in the space below AND at the bottom of each page of this booklet.

THIS IS AN OPEN BOOK, OPEN NOTES QUIZ.
NO PHONES, NO COMPUTERS, NO LAPTOPS, NO PDAS, ETC.

CIRCLE your recitation section number:

10:00	1. Madden/Hu	
11:00	2. Madden/Seater	3. Rinard/Ports
12:00	4. Rinard/Seater	5. Walfish/Ports
1:00	6. Walfish/Winstein	7. Katabi/Hu
2:00	8. Katabi/Winstein	

Do not write in the boxes below

1-3 (xx/20)	4-6 (xx/12)	7-10 (xx/30)	11-12 (xx/20)	13-14 (xx/18)	Total (xx/100)

Name:

I Reading Questions

1. [6 points]: Ethernet provides a medium access protocol (MAC) that allows multiple machines to communicate using a shared medium (Reading #9. Robert M. Metcalfe and David R. Boggs. Ethernet: Distributed packet switching for local computer networks) . TDMA is a different MAC protocol that has been presented in lectures. Say that there are n machines connected to the same bus. A TDMA approach would divide time into frames. Each frame has n slots. Each slot is sufficient to send one packet. Each one of the n machines is assigned a number between 1 and n. If machine i has a packet to send, it waits for slot i and sends its packet.

(Circle True or False for each choice.)

A. True / False Ethernet achieves a higher data rate than TDMA when a small number of machines have packets to send (i.e., much smaller than n), whereas TDMA is more efficient if most of the time all n machines have packets to send.

B. True / False Ethernet provides a guaranteed minimum throughput for each of the n machines

C. True / False Ethernet guarantees no collision because each machine senses the carrier before sending and does not send if another machine is currently using the medium.

2. [8 points]: Assume that an NFS (described in appendix 4.B) server contains a file /a/b and that an NFS client mounts the NFS server's root directory in the location /x, so that the client can now name the file as /x/a/b. Further assume that this is the only client and that the following code executes on the client:

```
chdir("/x/a"); // change to directory /x/a
rm("b");
```

The REMOVE message from the client to the server gets through and the server removes the file. Unfortunately, the response from the server to the client is lost so the client resends the message to remove the (now non-existent) file. The server receives the resent message. What happens next depends on the server implementation. Which of the following are correct statements?

(Circle True or False for each choice.)

A. True / False If the server maintains an in-memory reply cache in which it records all operations it previously executed, and there are no server failures, the server will return "OK".

B. True / False If the server maintains an in-memory reply cache but the server has failed, restarted, and its reply cache is empty, both of the following responses are possible: the server may return "file not found" or "OK".

C. True / False If the server is stateless, it will return "file not found".

D. True / False Because REMOVE is an idempotent operation, any server implementation will return "OK".

Name:

3. [6 points]: Ben Bitdiddle is performing an important *deterministic computation* that will take roughly 2 years to complete. In order to add some fault tolerance, he buys 30 identical computers and runs the same program with the same inputs on all of them. When they return their results after computing for 2 years, the voter selects the majority answer (voting is described in Chapter 8). Which of the following failures can this scheme tolerate, assuming the voter works correctly?

(Circle True or False for each choice.)

A. True / False The software carrying out the deterministic computation might have a bug in it, which causes it to compute the wrong answer for certain inputs.

B. True / False Cosmic rays might corrupt data stored in memory at one or two computers, causing them to return incorrect results.

Consider the following protocol: The sender assigns each packet a sequence number. The receiver sends cumulative acks, i.e, whenever it receives a packet, the receiver sends an ack for the next expected sequence number. For example, when it receives packets 1, 2, 4, the receiver sends acks 2, 3, 3 respectively. The sender uses a sliding window protocol. The size of the sliding window is *fixed* to 4 packets (because the receiver cannot buffer more than 4 packets at once). The network is given below. The data rate between the sender's computer and router is 200 packets/second, and the rate between router and receiver's computer is 100 packets/second. The one-way propagation delay between sender and receiver is 0.05 second (i.e., the round trip propagation delay is 0.1 second).

```
Sender --------------------- Router ---------------- Receiver
        200 packets/second          100 packets/second
```

4. [2 points]: Given the above protocol and network, what is the maximum number of packets the sender can send in a burst without waiting for acks?

5. [4 points]: Given the above protocol and network, what is the maximum send rate measured in packets per second?

6. [6 points]: Assume that at time t=0, the sender sent packets 1,2,3,4. By time t=0.15s, the sender has received acks 2,2,2 and nothing more. Which of the following statements are correct?

(Circle True or False for each choice.)

A. True / False The receiver has not received packet 4

B. True / False The sender is sure that packet 1 has been received.

C. True / False By t=0.15, the sliding window has advanced by 3 packets

Name:

II SURETHING

Alyssa P. Hacker decides to offer her own content delivery system, SURETHING. A SURETHING system contains 1000 computers that communicate over the Internet. Each computer has a unique ID, and they are thought of as (logically) being organized in a ring (see Figure 1). Each computer has *successors* as shown in the figure. The ring "wraps-around": the immediate successor of the computer with the highest ID (computer 251 in the figure) is the computer with the lowest ID (computer 8).

Each content item also has a unique ID, c, and the content will be stored at c's *immediate successor*: the first computer in the ring whose ID exceeds c (see Figure 1).

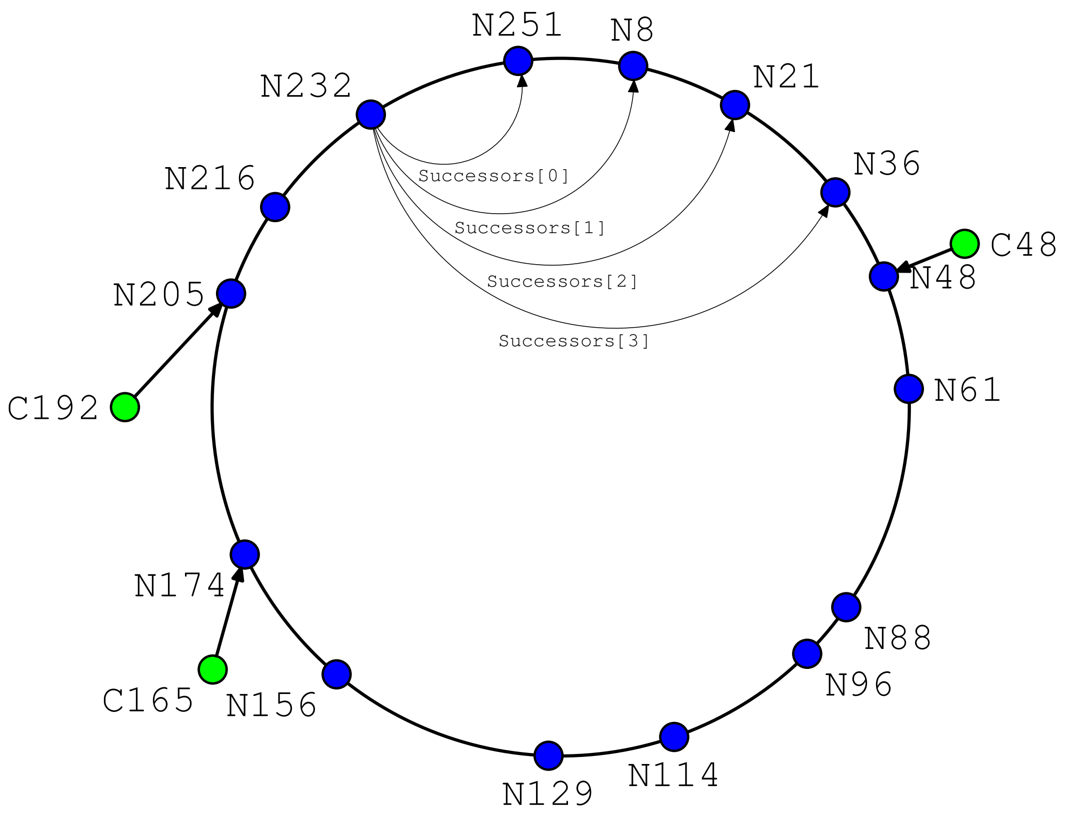


Figure 1: Arrangement of computers in a ring. Computer #232's pointers to its 4 successors are shown. The content item c with id #C192 will be stored at its immediate successor in the ring, computer #N205; item number #C48 will be stored at its immediate successor, computer #N48; and item number #C165 will be stored at its immediate successor, computer #N174.

Alyssa designs the system using two layers: a forwarding and routing layer (to find the IP address of the computer that stores the content) and a content layer (to store or retrieve the content).

Name:

II.1 Building a Forwarding and Routing Layer

Initially Alyssa decides that the routing step will work as follows: Each computer has a local table, *Successors*, that contains the ID and IP address of its 4 successors (the 4 computers whose IDs follow this computer's ID in the ring); the entries are ordered as they appear in the ring. These tables are set up when the system is initialized; you can assume they contain the right information. Unless a question states otherwise, you may assume that computers don't fail.

The forwarding and routing layer of each node provides a procedure GETLOC that can be called by the content layer to find the IP address of the immediate successor of some content item *c*. This procedure checks its local *Successors* table to see if it contains the immediate successor of the requested content; if not, it makes a remote procedure call to the GETLOCLOCAL procedure on the *most distant* successor in its *Successors* table. That computer returns the immediate successor of *c* if it is known locally in its *Successors* table; otherwise that node returns its *most distant* successor, and the originating computer continues the search there, iterating in this way until it locates *c*'s immediate successor.

For example, if computer N232 is looking for the immediate successor of *c* = C165 in the system shown in Figure 1, it will first look in its local table; since this table doesn't contain the immediate successor of *c*, it will request information from computer N36. Computer N36 also doesn't have the immediate successor of C165 in its local *Successors* table, and therefore it returns the IP address of computer N96. Computer N96 does have the immediate successor (computer N174) in its local *Successors* table and it returns this information. This sequence of RPCs is shown in Figure 2.

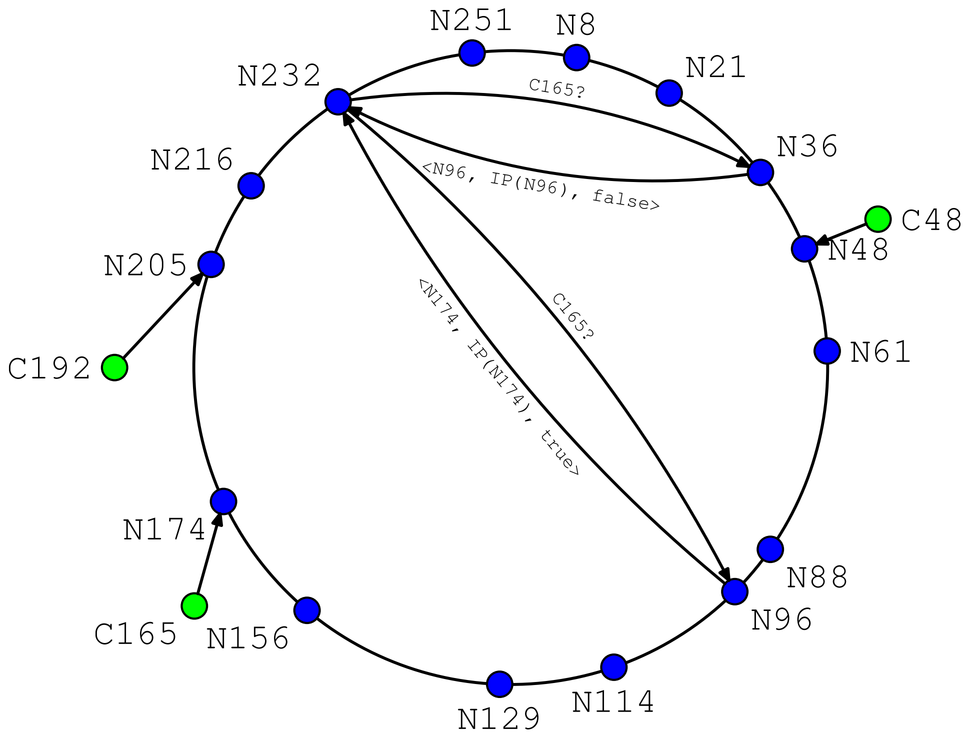


Figure 2: Sequence of RPCs and replies required for computer N232 to find the immediate successor of the content item with ID C165.

Name:

The code for GETLOC is given in Figure 3. This code correctly implements the approach described in the preceding paragraph. The code uses a helper procedure, LOOKUP, to search the *Successors* table; LOOKUP returns the first computer in the table that is a successor of the item of interest, or the last entry in the table, if the table doesn't contain a successor.

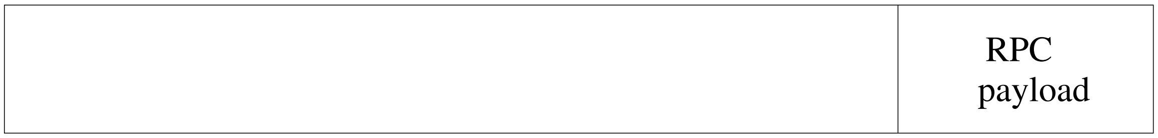
The code makes use of an RPC mechanism, invoked by DOCALL, that takes care of sending the message (including re-sending it if necessary), calling the requested procedure (GETLOCLOCAL) at the receiving end, getting the result from this call, and sending it back to the caller. The RPC mechanism runs at the application layer and uses UDP as the transport layer to actually do the communication.

```
structure pair { int id; IP loc; }
structure res { int id; IP loc; bool succ };
structure msg { int c; int id; IP loc; bool found; int resultCode}
define succsSz 4;
pair Successors[succsSz];

// the procedure called by the user
procedure GETLOC(int c)
 res ans ← LOOKUP(Successors, succsSz, c);     // check local table
 if ans.found then return ⟨ans.id, ans.loc⟩ ;     // answer found locally
 msg m, reply;
 m.c ← c;
 while TRUE do {     // search iteratively
  reply ← DOCALL(ans.loc, GETLOCLOCAL, m);
  if reply.resultCode = OK then {
   if reply.found then return ⟨reply.id, reply.loc⟩ ;     // remote computer found answer
   ans.loc ← reply.loc;     // remote computer returned its largest successor; continue searching there
  } else     // error handling (not shown)
 }


// the handler invoked by a RPC request when call is to GETLOCLOCAL
procedure GETLOCLOCAL(msg req)
 msg reply;
 res ans ← LOOKUP(Successors, succsSz, req.c);     // check local table
 reply.resultCode ← OK; reply.loc ← ans.loc; reply.id ← ans.id; reply.found ← ans.found;
 return reply;

procedure LOOKUP(pair[] table, int sz, int c)
 if c = myId then return ⟨myID, myIP, true⟩ ;     // check the local node
 for i ← 0 to sz − 1 do {
  if BETWEEN(c, myID, table[i].id) then return ⟨table[i].id, table[i].loc, true⟩ ;
     // between(c,x,y) returns true if c in (x, y] on the ring
 }
 return ⟨table[sz − 1].id, table[sz − 1].loc, false⟩ ;     // return most distant successor
```

Figure 3: Alyssa's initial implementation for GETLOC

Name:

7. [6 points]: Consider the packet corresponding to an RPC message when it is traversing an Ethernet link. What headers will be added? Draw a diagram of the packet with these headers. You do not need to indicate the specific contents of each part, but should label each header with the layer it belongs to.



8. [8 points]: While testing SURETHING, Alyssa notices that when the Internet attempts to deliver the RPC packets, they don't always arrive at their destination. Which of the following reasons might prevent a packet from arriving at its destination?

(Circle True or False for each choice.)

A. True / False A router discards the packet.

B. True / False The packet is corrupted in transit.

C. True / False The payload of the message contains the wrong *loc*

D. True / False The packet gets into a forwarding loop in the network.

Name:

For the next two questions, remember that computers don't fail and that all tables are initialized correctly.

9. [6 points]: Assume that c is an id whose immediate successor is not present in *Successors*, and n is the number of computers in the system. In the *best case*, how many remote lookups are needed before GETLOC(c) returns?

(Circle the BEST answer)

A. 0

B. 1

C. 2

D. O(log n)

E. O(n)

F. O(n^2)

10. [6 points]: Assume that c is an id whose immediate successor is not present in *Successors*, and n is the number of computers in the system. In the *worst case*, how many remote lookups are needed before GETLOC(c) returns?

(Circle the BEST answer)

A. 0

B. 1

C. 2

D. O(log n)

E. O(n)

F. O(n^2)

Name:

II.2 Building the Content Layer

Having built the forwarding and routing layer, Alyssa turns to building a content layer. At a high level, the system supports storing data that has an ID associated with it. Specifically, it supports two operations:

- PUT(c, $content$) stores $content$ in the system with ID c.
- GET(c) returns the content that was stored with ID c.

You can assume that content IDs are integers that can be used as arguments to GETLOC. (In practice, this can be assured by using a hash function that maps human-readable names to integers.)

Alyssa implements the content layer by using the forwarding and routing layer to choose which computers to use to store the content. For reliability, she decides to store every piece of content on two computers: the two immediate successors of the content's ID. She modifies GETLOC to return both successors, calling the new version GETLOC2. For example, if GETLOC2 is asked to find the successors of the content item with ID C165, it returns the IP addresses of computers N174 and N205 (given the information shown in Figure 1).

Once the correct computers are located using the forwarding and routing layer, Alyssa's implementation sends a REMOTEPUT RPC (which invokes the procedure REMOTEPUT on the remote computer) to each of these computers, requesting that the remote computer store the content in a file on its disk. To retrieve the content associated with a given ID, it sends a REMOTEGET RPC (which invokes the procedure REMOTEGET on the remote computer), requesting that the computer load the appropriate file from disk, if it exists, and return its contents.

The code for Alyssa's of GET and PUT is shown in Figure 4. It is not necessary to examine this code very closely, and you may assume it is correct. As before, DOCALL sends an RPC message, using retransmissions as necessary to ensure that the message is delivered. FETCH and STORE are an interface to the local file system; they maintain files so that previously stored content can be found, and return error codes when there are problems, e.g., if the requested content is not present. REMOTEGET and REMOTEPUT are't shown in the figure; you can assume they use FETCH and STORE to obtain or store the requested content.

11. [8 points]: What are the end-to-end properties of the content layer, assuming the algorithm in Figure 4? Assume that there are no failures of computers or disks while the system is running and that all tables are initialized correctly.

(Circle True or False for each choice.)

A. True / False GET(c) always returns the same content that was stored with ID c.

B. True / False PUT(c, $content$) stores the content at the two immediate successors of c.

C. True / False GET returns the content from the immediate successor of c.

D. True / False If the content has been stored on some computer, GET will find it.

Name:

```
**structure** $res$ { **int** $id$; **IP** $loc$; **bool** $succ$ };
**structure** $cmsg$ { **int** $c$; **data** $content$; **int** $resultcode$; }

// called by a user PUT request
**procedure** PUT(**int** $c$, **data** $content$)
  **cmsg** $m$;  $m.c \leftarrow c$;  $m.content \leftarrow content$;
  **res** $\langle ans1,\ ans2 \rangle \leftarrow$ GETLOC2($c$)
  **for** $ans$ **in** $\langle ans1,\ ans2 \rangle$ **do** {    // add to each of the two successors
    **if** ($ans.id$ = myID) **then** STORE($c$, $content$)
    **else** DOCALL($ans.loc$, REMOTEPUT, $m$);
  }

// called by a user GET request
**procedure** GET(**int** $c$)
  **cmsg** $m$;  $m.c \leftarrow c$;
  **cmsg** $reply$;
  **res** $\langle ans1,\ ans2 \rangle \leftarrow$ GETLOC2($c$);
  **if** $ans1.id$ = myID **or** $ans2.id$ = myID **then return** FETCH($c$);    // fetch from local disk if possible
  $reply \leftarrow$ DOCALL($ans1.loc$, REMOTEGET, $m$);
  **if** $reply.resultCode$ = OK **then return** $reply.content$;
      // first successor failed; try the second successor
  $reply \leftarrow$ DOCALL($ans2.loc$, REMOTEGET, $m$);
  **if** $reply.replycode$ = OK **then return** $reply.content$;
      // both successors failed; throw an error
```

Figure 4: Implementation of the content layer

Name:

12. [12 points]: Now, suppose that there are failures while the system is running. Which of the following properties of the content layer are true?

(Circle True or False for each choice.)

A. True / False One of the computers returned by GETLOC2 might not answer the GET or PUT call.

B. True / False PUT will sometimes be unable to store the content at the content's two immediate successors.

C. True / False GET will successfully return the requested content, assuming it was stored previously.

D. True / False If one of the two computers on which PUT the content is not failed when GET runs, GET will succeed in retrieving the content.

Name:

II.3 Improving Forwarding Performance

In this question we return to the forwarding and routing layer; you can ignore the content layer.

Alyssa isn't very happy with the performance of the system, in particular GETLOC. Her friend Lem E. Tweakit suggests the following change: each computer maintains a *NodeCache*, which contains information about the IDs and IP addresses of computers in the system. The *NodeCache* table initially contains information about the computers in *Successors*. The modified code for GETLOC is in Figure 5. You need not examine this code closely. It differs from the code in Figure 3 in only two ways: (1) it looks for the local answer in the *NodeCache* rather than in *Successors*, and (2) it inserts the new pair into the *NodeCache*. The changes are marked on the figure.

For example, initially the *NodeCache* at computer N232 contains entries for computers N251, N8, N21, N36, given the setup in Figure 1. But after computer N232 communicates with computer N36 and learns the ID and IP address of computer N96, its *NodeCache* now contains entries for computers N251, N8, N21, N36, and N96.

This code uses the same LOOKUP procedure shown in Figure 3. INSERT adds the new $\langle id,\ loc\rangle$ pair to the *NodeCache* if it isn't already present, and it maintains the entries in the *NodeCache* in ring order; you may assume it does this correctly.

Name:

```
structure pair { int id; IP loc; }
structure res { int id; IP loc; bool succ };
structure msg { int c; int id; IP loc; bool found; int resultcode}
define succsSz 4;
pair Successors[succsSz];
pair NodeCache[1000];   // initially contains all entries in Successors
int NodeCacheSize;   // number of entries in NodeCache; initially equals succsSz

// the procedure called by the user
procedure GETLOC(int c)
  res ans ← LOOKUP(NodeCache, NodeCacheSize, c);   // check local NodeCache; for LOOKUP see Figure 3
  if ans.found then return ⟨ans.id, ans.loc⟩ ;   // answer found locally
  msg m, reply;
  m.c ← c;
  while TRUE do {   // search iteratively
    reply ← DOCALL(ans.loc, GETLOCLOCAL, m);
    if reply.resultCode = OK then {
      INSERT(NodeCache, ⟨reply.id, reply.loc⟩);   // add response to NodeCache
      if reply.found then return ⟨reply.id, reply.loc⟩ ;   // remote computer found answer
      ans.loc ← reply.loc;   // remote computer returned its largest successor; continue searching there
    } else   // error handling (not shown)
  }

// the handler invoked by a RPC request
procedure GETLOCLOCAL(msg req)
  msg reply;
  res ans ← LOOKUP(NodeCache, NodeCacheSize, c);   // check local NodeCache; for LOOKUP see Figure 3
  reply.resultCode ← OK; reply.loc ← ans.loc; reply.id ← ans.id; reply.found ← ans.found;
  return reply;
```

Figure 5: Lem's implementation ofGETLOC with caching

Name:

13. [6 points]: Assume that c is a content ID whose immediate successor is not one of the computers listed in *Successors*, and n is the number of computers in the system. In the *best case*, how many remote lookups are needed before GETLOC(c) returns?

(Circle the BEST answer)

A. 0

B. 1

C. 2

D. O(log n)

E. O(n)

F. O(n^2)

14. [12 points]: Alyssa is wondering if the implementation suggested by Lem is correct. Which of the following statements are true about the implementation of figure 5, assuming no failures.

(Circle True or False for each choice.)

A. True / False If the immediate successor of c is present in *NodeCache*, GETLOCLOCAL will return this computer's address (*loc*).

B. True / False If the immediate successor of c is not present in *NodeCache*, the reply message from GETLOCLOCAL will contain **found = false**.

C. True / False If the immediate successor of c is present in *NodeCache*, GETLOC will return this computer's address.

D. True / False GETLOC will always return the immediate successor of c.

End of Quiz II

Name: