Started on	Thursday, 9 September 2021, 8:55 AM
State	Finished
Completed on	Tuesday, 14 September 2021, 4:27 PM
Time taken	5 days 7 hours
Grade	Not yet graded

Information

This is the 'electronic' part of the COSC 264 mid-term test, worth 85% of the overall marks. A few important hints:

- The programming language in the Coderunner problems is Python3.
- Please read text and instructions carefully, be careful with units.
- In the CodeRunner questions you will see tests of the form '(abs(theFunctionToWrite(..) - someNumber)< threshold)'. In tests like this the expected response in the test case is 'someNumber', but since floating point arithmetic is not exact, we allow for an error of up to 'threshold'. The abs(x) function returns the magnitude of its argument.

Question **1**

Correct

Mark 2.00 out of 2.00

Please calculate the propagation delay for a signal traveling a distance of 15,000 km, assuming a speed of light of C=300,000 km/s. Please give your answer in seconds.

Answer: 0.05 ✓

Correct

Marks for this submission: 2.00/2.00.

Question **2**

Correct

Mark 2.00 out of 2.00

Please calculate the transmission delay for a packet of length *L*=1,500 bytes over a link with a data rate of *R*=10 Mbps. Please give it in seconds.

Answer: 0.0012 ✓

Correct

Marks for this submission: 2.00/2.00.

Question **3**

Correct

Mark 2.40 out of 4.00

Suppose we transmit a packet of length *L* bits over a channel on which bit errors are statistically independent and happen with bit error probability *P*. Please find an expression for the probability that a received packet has at least one bit error and implement it in Python.

For example:

Test	Result
`print (abs(packeterrorprobability(1000, 0.001)-0.6323)<0.0001)`	True

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```python
def packeterrorprobability (pktLength_b, bitErrorProb):
    L = pktLength_b
    P = bitErrorProb
    return (1 - (1-bitErrorProb)** pktLength_b)
```

	Test	Expected	Got	
✓	`print (abs(packeterrorprobability(1000, 0.001)-0.6323)<0.0001)`	True	True	✓
✓	`print (abs(packeterrorprobability(1000, 0.0001)-0.0951)<0.0001)`	True	True	✓

Passed all tests! ✓

Correct

Marks for this submission: 4.00/4.00. Accounting for previous tries, this gives **2.40/4.00**.

Question **4**

Correct

Mark 2.00 out of 2.00

Use your function from the previous question to calculate the probability of at least one bit error when the packet length is *L*=2,000 bits and the bit error rate is *P*=0.0001. Please give three digits after the decimal point, no rounding.

Answer: 0.181 ✓

Correct

Marks for this submission: 2.00/2.00.

Question **5**

Correct

Mark 0.00 out of 6.00

Suppose that through error-correction coding we have the ability to correct one wrong bit in a packet of *L* bits in total. To be erroneous, such a packet would need to have at least two bit errors. Please find an expression for the probability that a packet of length *L* bits has at least two bit errors (assuming bit errors are independent and occur with bit error probability *P*) and implement it as a Python function.

For example:

Test	Result
`print (abs(twowrongbits(1000, 0.001)-0.2642)<0.0001)`	True

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```python
import math
def twowrongbits (pktLength_b, bitErrorProb):
    L = pktLength_b
    P = bitErrorProb
    q = 1 - bitErrorProb
    no_error = q**pktLength_b
    one_error = (math.factorial(L) / (math.factorial(L-1))) * (bitErrorProb) * (q**(L-1))
    return 1 - no_error - one_error
```

	Test	Expected	Got	
✓	`print (abs(twowrongbits(1000, 0.001)-0.2642)<0.0001)`	True	True	✓
✓	`print (abs(twowrongbits(1000, 0.0005)-0.0901)<0.0001)`	True	True	✓
✓	`print (abs(twowrongbits(3000, 0.0003)-0.2275)<0.0001)`	True	True	✓

Passed all tests! ✓

Correct

Marks for this submission: 6.00/6.00. Accounting for previous tries, this gives **0.00/6.00**.

Information

We consider **circuit switching**.

We are given a system with a number of $N+2$ stations A, S_1, S_2, ..., S_N, B such that end host A is connected to the first switch S_1, the first switch S_1 is connected to switch S_2, and so on, and the last switch S_N is connected to the other end host B (i.e. all the stations form a chain). A wants to establish a circuit to B, which has to go through all the N intermediate switches.

We want to calculate the overall call-setup-delay, i.e. in the time it takes to go through the "unproductive" connection setup phase, which we have to complete before we can transmit any data. All the links in the system have the same length of L km, the speed of light on the cables is C km/s. The data rate supported on all links is R bps, and there are no transmission errors on the links.

To establish a circuit, station A will send a particular message, the **call-setup-request** message of M_{req} bits length to the first switch S_1. After receiving this message, switch S_1 will need a time of P s to process it, before S_1 continues to send the same message further on to switch S_2. This way the message travels through all the switches and finally reaches end host B. Once B has fully received the message, it will process it (which again takes P s) and then instantaneously generate a **call-setup-response** message of M_{resp} bits length, which it sends back to A (through all the switches S_1 to S_N). After switch S_N has fully received the call-setup-response message, it will process it (taking P s) and forward it to switch S_{N-1} and so on. After A has completely received the call-setup-response message, A will process it (which takes P s) and after that station A can commence with actual data transmission.

In the following few questions you are asked to develop mathematical expressions for different components of the total time needed until A can commence data transmission, and to implement these in Python.

Question **6**

Correct

Mark 3.00 out of 3.00

Please work out a general expression for the total combined propagation delay that all call-setup-request and call-setup-response messages cause, and implement it as a Python function.

For example:

Test	Result
`print (abs(propagation_delay(3, 7500, 200000)-0.3)<0.0001)`	`True`

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```python
def propagation_delay (numberSwitches, cableLength_km, speedOfLight_kms):
    N     = numberSwitches
    L     = cableLength_km
    C     = speedOfLight_kms
    return ((cableLength_km/speedOfLight_kms) * (numberSwitches + 1)) * 2
```

	Test	Expected	Got	
✓	`print (abs(propagation_delay(3, 7500, 200000)-0.3)<0.0001)`	`True`	`True`	✓
✓	`print (abs(propagation_delay(5, 7500, 200000)-0.4499)<0.0001)`	`True`	`True`	✓
✓	`print (abs(propagation_delay(3, 10000, 200000)-0.4)<0.0001)`	`True`	`True`	✓

Passed all tests! ✓

Correct

Marks for this submission: 3.00/3.00.

Question **7**

Correct

Mark 3.00 out of 3.00

Please work out a general expression for the total combined transmission delay of all call-setup-request and call-setup-response transmissions and implement it as a Python function.

For example:

Test	Result
`print (abs(transmission_delay(3, 10000000, 2000, 1000)-0.0012)<0.0001)`	`True`

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
def transmission_delay (numberSwitches, dataRate_bps, messageLengthRequest_b, messageLengthResponse_b):
    N     = numberSwitches
    R     = dataRate_bps
    Mreq  = messageLengthRequest_b
    Mresp = messageLengthResponse_b
    return ((messageLengthRequest_b/dataRate_bps)*(numberSwitches+1) +
            ((messageLengthResponse_b/dataRate_bps)*(numberSwitches+1)))
```

	Test	Expected	Got	
✓	`print (abs(transmission_delay(3, 10000000, 2000, 1000)-0.0012)<0.0001)`	`True`	`True`	✓
✓	`print (abs(transmission_delay(3, 10000000, 3000, 2000)-0.002)<0.0001)`	`True`	`True`	✓

Passed all tests! ✓

Correct

Marks for this submission: 3.00/3.00.

Question **8**

Correct

Mark 3.00 out of 3.00

Please work out a general expression for the combined total processing delay incurred for the processing of all call-setup-request and call-setup-response transmissions, and implement it as a Python function.

For example:

Test	Result
`print (abs(processing_delay(3, 0.001)-0.008)<0.0001)`	`True`

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```python
def processing_delay (numberSwitches, processingTimes_s):
    N      = numberSwitches
    P      = processingTimes_s
    return ((numberSwitches+1) *processingTimes_s)* 2
```

	Test	Expected	Got	
✓	`print (abs(processing_delay(3, 0.001)-0.008)<0.0001)`	`True`	`True`	✓
✓	`print (abs(processing_delay(5, 0.001)-0.012)<0.0001)`	`True`	`True`	✓

Passed all tests! ✓

Correct

Marks for this submission: 3.00/3.00.

Question **9**

Correct

Mark 3.00 out of 3.00

Now combine your expressions from the previous problems to find a general expression for the call-setup delay (i.e. the time between A starting to transmit the call-setup-request message and A finishing receiving and processing the call-setup-response message) and implement it as a Python function.

For example:

Test	Result
`print (abs(connection_setup_delay(3, 7500, 200000, 10000000, 2000, 1000, 0.001)-0.3092)<0.0001)`	True

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
def connection_setup_delay (numberSwitches, cableLength_km, speedOfLight_kms, dataRate_bps, messageLengthRe
    N     = numberSwitches
    L     = cableLength_km
    C     = speedOfLight_kms
    R     = dataRate_bps
    Mreq  = messageLengthRequest_b
    Mresp = messageLengthResponse_b
    P     = processingTimes_s

    prog_delay = ((L/C)*(N+1))*2
    trans_delay = ((Mreq/R) * (N+1)) + ((Mresp/R) * (N+1))
    process_delay = ((numberSwitches + 1) * P) * 2

    return prog_delay + trans_delay + process_delay
```

	Test	Expected	Got	
✓	`print (abs(connection_setup_delay(3, 7500, 200000, 10000000, 2000, 1000, 0.001)-0.3092)<0.0001)`	True	True	✓
✓	`print (abs(connection_setup_delay(5, 7500, 200000, 10000000, 2000, 1000, 0.001)-0.4638)<0.0001)`	True	True	✓
✓	`print (abs(connection_setup_delay(3, 7500, 200000, 20000000, 2000, 1000, 0.001)-0.3086)<0.0001)`	True	True	✓
✓	`print (abs(connection_setup_delay(3, 10000, 200000, 20000000, 2000, 1000, 0.001)-0.4086)<0.0001)`	True	True	✓

Passed all tests! ✓

Correct

Marks for this submission: 3.00/3.00.

Question **10**

Correct

Mark 1.00 out of 1.00

Please evaluate your expression for a setup with N=10 switches, a link length of L=2,000 km, a speed of light on cables of C=200,000 km/s, a supported data rate of R=10,000,000 bps, a request message length of M_{req}=2,000 bits, a response message length of M_{resp}=1,000 bits and a processing delay of P=0.001 s.

Answer: 0.2453 ✓

Correct

Marks for this submission: 1.00/1.00.

Question **11**

Correct

Mark 1.00 out of 1.00

Which socket function do you need to call to link a socket to a specific port number / IP address?

Select one:

- a. select()
- [✓ b. bind()] ✓
- c. connect()
- d. socket()
- e. accept()

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

Question **12**

Correct

Mark 1.00 out of 1.00

In which order does a TCP server call these two functions?

Select one:

- a. accept(), listen()
- [✓ b. listen(), accept()] ✓

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

Question **13**

Correct

Mark 0.33 out of 1.00

Which socket function must a UDP client call before it can call write()?

Select one:

- a. accept()
- [✓ b. connect()] ✓
- c. bind()
- d. listen()
- e. recvfrom()

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00. Accounting for previous tries, this gives **0.33/1.00**.

Question **14**

Correct

Mark 1.00 out of 1.00

Which helper function do you need to call to convert a 16-bit integer (e.g. a port number) from the host representation to network representation?

Select one:

- [✓ a. htons()] ✓
- b. ntohs()
- c. htonl()
- d. ntohl()

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

Question **15**

Correct

Mark 1.00 out of 1.00

How many sockets can be bound to one particular IP-address / port number combination, e.g. in a TCP server?

Select one:

- a. Two
- b. One
- [✓ c. Arbitrarily many] ✓
- d. Five

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

Question **16**

Correct

Mark 2.00 out of 2.00

Which of the seven layers in the OSI reference model is responsible for ensuring end-to-end reliable, in-sequence transfer?

Select one:

- a. Link layer
- b. Representation layer
- c. Session layer
- d. Network layer
- e. Physical layer
- [✓ f. Transport layer] ✓
- g. Application layer

Your answer is correct.

Correct

Marks for this submission: 2.00/2.00.

Question **17**

Complete

Marked out of 4.00

Please explain briefly why error control is needed on the transport layer in a multihop network, even if all the link layer protocols in the network operate with perfect reliability.

The sending side sends segments are divided into multiple packets at the network layer and each packet on multiple frames at the link level.

Each segment travels the network (divided as frames and packets) and is recomposed only at the receiving side.

Between the sending and receiving side could be a lot of intermediate routers. During that transit, there could be problems as:

One or more frames are discarded.

One or more packets get lost.

Packets lose their original order.

A malfunctioning router modifies the data in a packet

These problems will pass undetected through the routers until they get to the error control of the transport layer on the receiving side.

This error control verifies that there weren't problems on the underneath layers problems and asks for retransmission.

Question **18**

Correct

Mark 2.00 out of 2.00

Suppose we have a protocol on some layer N. The N-protocol uses sequence numbers in the header of N-PDUs, and the sequence number header field has a width of four bits. Suppose that a particular packet has sequence number 15, and the transmitter infers that the packet needs to be re-transmitted. Which sequence number will the re-transmission packet have?

Answer: 15 ✓

Correct

Marks for this submission: 2.00/2.00.

Question **19**

Correct

Mark 2.00 out of 2.00

A layer-N protocol offers a fragmentation-and-reassembly mechanism in which each fragment can carry 200 bytes of payload data. The header size of an N-PDU (layer N-packet) is 80 bytes. The total size of a fragment is given by the header size and the size of the payload.

The N-layer protocol entity is given a message from the higher layers with a total of 1024 bytes. What is the total size of the last fragment?

Answer: 104 ✓

Correct

Marks for this submission: 2.00/2.00.

Question **20**

Correct

Mark 2.00 out of 2.00

A layer-N protocol offers a fragmentation-and-reassembly mechanism in which each fragment can carry 200 bytes of payload data. The header size of an N-PDU (layer N-packet) is 80 bytes.

The N-layer protocol entity is given a message from the higher layers with a total of 1024 bytes. How many fragments are minimally needed?

Answer: 6 ✓

Correct

Marks for this submission: 2.00/2.00.

Question **21**

Complete

Marked out of 5.00

Please explain the operation of FDMA.

FDMA stands for Frequency division medium access

FDMA requires N receivers for every N station or tunable receiver.

Each station is assigned a frequency from the available bandwidth [B/S]

FDMA supports collision-free parallel transmission as each station has a dedicated frequency.

PROS:

FDMA is good for CBR and VBR

N stations can transmit in parallel

No need for time synchronization between transmitters

CONS:

Need N transmitters

frequency synchronization is required

No re-use of unused frequencies.

Question **22**

Complete

Marked out of 5.00

Please explain the operation of the Ethernet MAC protocol, without the details of the backoff function.

MAC protocols are rules by which distributed stations coordinate access to a common channel to share it efficiently and in a manner satisfying given performance requirements.

MAC protocols are schemes that allow a number of users to coordinate access to a common channel.

The MAC layer is often regarded as a separate sub-layer between PHY and link-layer.

This view is supported by the fact that the MAC has a distinguished task not covered by any other layer.

MAC protocols are heavily influenced by the properties of the underlying transmission medium.

The shared channel is a broadcast medium, i.e. transmission of one station is heard by all other stations, not necessarily true for wireless transmission media.

In the case of parallel transmissions all contending transmissions are garbled, i.e. cannot be reliably decoded, not necessarily true for wireless transmission media Often not true for CDMA systems, also not in OFDMA.

Question **23**

Correct

Mark 2.70 out of 3.00

Please write a Python3 function which returns the upper bound (i.e. the largest allowed value) of the Ethernet backoff window interval depending on the number of collisions.

For example:

Test	Result
`print (backoff(3))`	7

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
def backoff (numColl):
    if numColl > 10:
        numColl = 10
    return (2**numColl) - 1
```

	Test	Expected	Got	
✓	`print (backoff(3))`	7	7	✓
✓	`print (backoff(9))`	511	511	✓
✓	`print (backoff(10))`	1023	1023	✓
✓	`print (backoff(11))`	1023	1023	✓

Passed all tests! ✓

Correct

Marks for this submission: 3.00/3.00. Accounting for previous tries, this gives **2.70/3.00**.

Question **24**

Correct

Mark 4.50 out of 5.00

This question is not related to any specific MAC protocol we have discussed.

Suppose we have a system with *N* available time slots, and we have two stations. Each time slot is sufficient for one packet and each station picks one of the *N* time slots with uniform probability, independent of the other station. Find an expression for the probability that the two stations pick the same slot (i.e. their packets collide) and implement it as a Python function.

For example:

Test	Result
`print ("{:.4f}".format(collprob(2)))`	`0.5000`

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```python
def collprob (numSlots):
    N = numSlots
    return 1 / numSlots
```

	Test	Expected	Got	
✓	`print ("{:.4f}".format(collprob(2)))`	`0.5000`	`0.5000`	✓
✓	`print ("{:.4f}".format(collprob(3)))`	`0.3333`	`0.3333`	✓
✓	`print ("{:.4f}".format(collprob(4)))`	`0.2500`	`0.2500`	✓
✓	`print ("{:.4f}".format(collprob(5)))`	`0.2000`	`0.2000`	✓

Passed all tests! ✓

Correct

Marks for this submission: 5.00/5.00. Accounting for previous tries, this gives **4.50/5.00**.

Question **25**

Correct

Mark 2.00 out of 4.00

Please implement a Python function which checks whether a 32-bit IP destination address '*dst*' matches an entry *a.b.c.d/k* in the forwarding table, where *a.b.c.d* is given simply as a 32-bit number (*a.b.c.d* is called '*netaddr*' in the parameter list) and the '*kbitmask*' parameter is a 32-bit /k network mask. The function should return True if the destination address matches the entry and False if not.

For your own testing, you can copy+paste the following list of all bitmasks:

```
bitmasks = [0b00000000000000000000000000000000,
           0b10000000000000000000000000000000,
           0b11000000000000000000000000000000,
           0b11100000000000000000000000000000,
           0b11110000000000000000000000000000,
           0b11111000000000000000000000000000,
           0b11111100000000000000000000000000,
           0b11111110000000000000000000000000,
           0b11111111000000000000000000000000,
           0b11111111100000000000000000000000,
           0b11111111110000000000000000000000,
           0b11111111111000000000000000000000,
           0b11111111111100000000000000000000,
           0b11111111111110000000000000000000,
           0b11111111111111000000000000000000,
           0b11111111111111100000000000000000,
           0b11111111111111110000000000000000,
           0b11111111111111111000000000000000,
           0b11111111111111111100000000000000,
           0b11111111111111111110000000000000,
           0b11111111111111111111000000000000,
           0b11111111111111111111100000000000,
           0b11111111111111111111110000000000,
           0b11111111111111111111111000000000,
           0b11111111111111111111111100000000,
           0b11111111111111111111111110000000,
           0b11111111111111111111111111000000,
           0b11111111111111111111111111100000,
           0b11111111111111111111111111110000,
           0b11111111111111111111111111111000,
           0b11111111111111111111111111111100,
           0b11111111111111111111111111111110,
           0b11111111111111111111111111111111]
```

The test cases below also make use of a private function 'ip2Int', which for your convenience is given here as well:

```
def ip2Int (dd):
   digits=dd.split('.')
   intIp=0
```

```
    cnt=0
    for num in reversed(digits):
        intIp += int(num) * 256 **(cnt)
        cnt +=1
    return intIp
```

For example:

Test	Result
`print (match(ip2Int("130.149.49.77"), ip2Int("130.149.0.0"), bitmasks[16]))`	`True`

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
def match (dst, netaddr, kbitmask):
    bin_and = dst & kbitmask
    first = (bin_and & 0b11111111000000000000000000000000) >> 24
    second = (bin_and & 0b00000000111111110000000000000000) >> 16
    third = (bin_and & 0b00000000000000001111111100000000) >> 8
    fourth = (bin_and & 0b00000000000000000000000011111111)

    d_first = (netaddr & 0b11111111000000000000000000000000) >> 24
    d_second = (netaddr & 0b00000000111111110000000000000000) >> 16
    d_third = (netaddr & 0b00000000000000001111111100000000) >> 8
    d_fourth = (netaddr & 0b00000000000000000000000011111111)
    result = [first, second, third, fourth]
    d_result = [d_first, d_second, d_third, d_fourth]

    for i in range(4):
        if result[i] != d_result[i]:
            return False

    return True
```

	Test	Expected	Got	
✓	`print (match(ip2Int("130.149.49.77"), ip2Int("130.149.0.0"), bitmasks[16]))`	`True`	`True`	✓
✓	`print (match(ip2Int("130.149.49.77"), ip2Int("130.149.0.0"), bitmasks[17]))`	`True`	`True`	✓
✓	`print (match(ip2Int("130.149.49.77"), ip2Int("130.149.0.0"), bitmasks[18]))`	`True`	`True`	✓
✓	`print (match(ip2Int("130.149.49.77"), ip2Int("130.149.0.0"), 19))`	`False`	`False`	✓

Passed all tests! ✓

Correct

Marks for this submission: 4.00/4.00. Accounting for previous tries, this gives **2.00/4.00**.

Question **26**

Correct

Mark 3.00 out of 3.00

Suppose an IP router has the following forwarding table (there are no further entries and in particular no default entry):

Destination Network / Netmask	Outgoing interface
135.2.0.0 / 16	eth0
135.6.10.0 / 24	eth1
136.4.12.0 / 24	eth2
137.6.0.0 / 16	eth0
132.16.12.0 / 24	directly attached

Please identify the forwarding decisions that the router makes for the following destination addresses.

135.2.33.55	Forward to eth0	✓
136.4.12.13	Forward to eth2	✓
135.0.0.0	Drop	✓
132.16.12.6	Deliver to directly attached network	✓
135.1.0.0	Drop	✓
135.6.10.15	Forward to eth1	✓

Your answer is correct.

Correct

Marks for this submission: 3.00/3.00.

Question **27**

Correct

Mark 2.00 out of 3.00

In real IPv4 networks it can happen that a forwarding table can have several entries matching the same destination address. For example, there can be two entries as follows:

Destination Network / Netmask	**Outgoing interface**
130.149.0.0 / 16	eth1
130.149.49.0 / 24	eth2

In such a case a packet to destination address 130.149.49.77 would match both entries, and the IP protocol would choose the **more specific** entry, i.e. the entry with the larger netmask (here: it would choose the /24 entry over the /16 entry).

Please determine the decisions that a router will make for the following destination addresses, assuming that the forwarding table is just the table given here, with no further entries (and particularly no default entry).

130.148.49.13	Drop	✓
130.149.49.13	Forward to eth2	✓
130.149.50.13	Forward to eth1	✓

Your answer is correct.

Correct

Marks for this submission: 3.00/3.00. Accounting for previous tries, this gives **2.00/3.00**.

Question **28**

Complete

Marked out of 5.00

Describe briefly the steps that an IP router performs for an incoming IP packet (without detail about the forwarding table lookup operation).

IP Routing is mostly concerned with networks, i.e. forwarding tables in routers mostly store <network-id>'s – it is the responsibility of the last router on a path to deliver an IP datagram to a directly connected host.

Incoming packets are checked for correctness and stored in IP input queue – correctness includes:

right value in IP version field

correct IP header checksum

Next, it is checked if the packet is destined to this host/router or to the broadcast address of any network this host/router is directly attached to.

If so, protocol demultiplexing is carried out:

The Protocol field in the IP header is checked for its value

The packet payload is delivered to the software entity implementing the indicated higher-layer protocol

The packet is not processed any further!

Question **29**

Correct

Mark 2.00 out of 2.00

What is the minimum size of the IPv4 header (in bytes)?

Answer: 20 ✓

Correct

Marks for this submission: 2.00/2.00.

Question **30**

Correct

Mark 2.00 out of 2.00

Consider the following statement: "The TLL mechanism of IPv4 prevents routing loops from happening."

True or False?

Select one:

○ True

[✓ False] ✓

Correct

Marks for this submission: 2.00/2.00.

Question **31**

Correct

Mark 2.00 out of 2.00

We are given the IP address 130.149.49.77 and we know that this address belongs to a /24 network. Please give the host part of this address as a decimal number.

Answer: 77 ✓

Correct

Marks for this submission: 2.00/2.00.

◄ Quiz: Layering and reference models, Basic protocol mechanisms (Practice copy)

Jump to...

Quiz: Local area networks and medium access control ►