CS 168 Spring 2025 — Introduction to the Internet Midterm

Solutions last updated: Thu, March 13, 2025

Print Your Name: _____

Print Your Student ID: _____

You have 110 minutes. There are 7 questions of varying credit. (100 points total)

Question:	1	2	3	4	5	6	7	Total
Points:	12	14	12	10	10	16	26	100

For questions with **circular bubbles**, you may select only one choice.

- ○ Unselected option (Completely unfilled)
- ◎ Don't do this (it will be graded as incorrect)
- ● Only one selected option (completely filled)

For questions with **square checkboxes**, you may select one or more choices.

- ■ You can select
- ■ multiple squares
- ☑ Don't do this (it will be graded as incorrect)

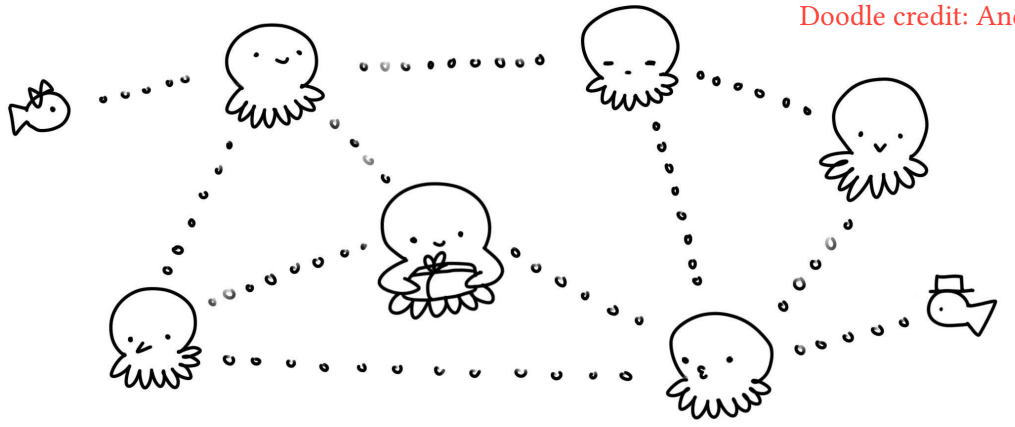
Anything you write outside the answer boxes or you ~~cross out~~ will not be graded. If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we will grade the worst interpretation.

Honor Code: Read the honor code below and sign your name.

> I understand that I may not collaborate with anyone else on this exam, or cheat in any way. I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that academic misconduct will be reported to the Center for Student Conduct and may further result in, at minimum, negative points on the exam.

Sign your name: _____

Doodle credit: Andrea Lou



Clarifications

- In Q2, z > x.
- In Q3, advertisements are sent and received on the same time step, regardless of link cost. For example, if R3 sends an advertisement at t=10, R5 receives the advertisement at t=10.
- In Q3, “at each subsequent time step” means that the first advertisements are sent at t=2, not t=1.
- In Q2.4, consider only packets of the same size as the packet shown in the diagram.

Q1 *Potpourri* **(12 points)**

Q1.1 (2 points) When reading a UDP-over-IP packet, which of these describes the byte immediately after the end of the Layer 3 header? Select all that apply.

- [x] Layer 4 header
- [ ] Layer 7 header
- [x] Layer 3 payload
- [ ] Layer 4 payload
- [ ] Layer 7 payload
- ○ None of the above

Solution:

Layer 4 header:

Recall that lower-layer headers are wrapped around higher-layer headers, so when reading the bytes of a packet, the lower-layer headers (outermost) appear before the higher-layer headers (inner).

Therefore, in a UDP-over-IP packet, the Layer 3 header is immediately followed by the Layer 4 header.

Layer 3 payload:

Recall that from the Layer 3 perspective, everything after the header is the payload.

Therefore, the byte following the Layer 3 header can also be thought of as the Layer 3 payload.

This byte is part of the Layer 4 header, so it cannot be considered part of the Layer 4 payload.

Q1.2 (1 point) If only end hosts implement reliability, it is possible to guarantee reliability.

- ● True
- ○ False

Solution: True. The end-to-end principle says that end hosts alone are sufficient to guarantee reliability.

Q1.3 (1 point) If only routers implement reliability, it is possible to guarantee reliability.

- ○ True
- ● False

Solution: False. The end-to-end principle says that routers cannot guarantee reliability by themselves.

Q1.4 (1 point) If both routers and end hosts implement reliability, it is possible to guarantee reliability.

- ● True
- ○ False

Solution: True. The end-to-end principle says that end hosts alone are sufficient to guarantee reliability. If the routers also implement reliability, we'd still have guaranteed reliability, just from the end hosts.

Q1.5 (1 point) In a circuit-switched network, routers can always use destination-based forwarding to forward packets.

○ True ● False

Solution: False. If two packets are part of two different reservations, then they might get forwarded along two different next-hops, even if their destination is the same.

Q1.6 (1 point) Circuit switching guarantees that packets are reliably delivered.

○ True ● False

Solution: False. Circuit switching is a design paradigm based on reserving resources, but it doesn't say anything about guaranteeing reliable packet delivery.

For example, if we implemented circuit switching at Layer 3, there might not be a guarantee of reliability if Layer 3 only offers a best-effort service model.

Q1.7 (1 point) If a router's packet queue is not empty, the router always processes packets in its queue before processing packets arriving from the incoming links.

○ True ● False

Solution: The intended answer was False. Whether or not the router sends a packet from its queue or from one of its incoming links is determined by a packet scheduling algorithm, which could vary which packet the router sends out first.

Grading: After the exam, some students pointed out that in Spring 2025, Lecture 8, Slide 29 says to "assume FIFO" for the order we send out packets. The slide's intention is to say "assume FIFO unless otherwise specified," but since the text of the slide is a little misleading, we gave students the benefit of the doubt and accepted True for full credit.

The bug in the slides will be fixed in future semesters, so that the intended answer (False) is the only correct answer.

Q1.8 (2 points) In Project 2 (routing), you implemented incremental updates in Stage 10 by supporting the `force=False` option in the `send_routes` function.

If we instead always called `send_routes` with `force=True`, would the code result in valid routing state?

- (●) Yes, but with more advertisements sent.
- ( ) No, too many advertisements get sent.
- ( ) Yes, but with fewer advertisements sent.
- ( ) No, not enough advertisements get sent.

Solution: Incremental updates act as a performance optimization by avoiding sending duplicate advertisements.

If we always disabled incremental updates, then every advertisement gets sent out of every link, even if the advertisement was already previously-sent.

This will lead to more advertisements sent, since some additional duplicate advertisements will be sent.

The resulting routing protocol is still correct and will result in valid routing state, because the duplicate advertisements will simply be ignored when they are received.

Q1.9 (1 point) In BGP, the size of a router's forwarding table scales with the number of hosts in the router's own AS, plus the number of other ASes.

- ( ) True
- (●) False

Solution: In BGP, ASes can aggregate prefixes, so we can use a single table entry to represent many external ASes.

In other words, it is possible to add many external ASes, without increasing the size of the forwarding table, as long as those ASes can be aggregated into a single table entry.

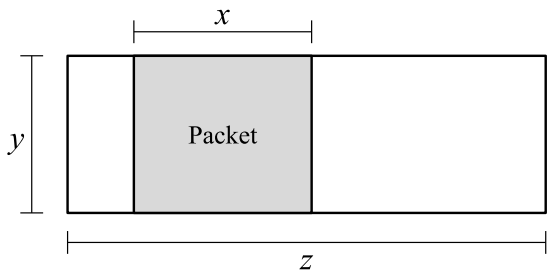
Q1.10 (1 point) The IPv4 checksum is used to detect payload corruption.

- ( ) True
- (●) False

Solution: Since the IPv4 checksum is only computed over the packet header, it can't be used to catch payload corruption.

Q2 *Pipes* **(14 points)**

Consider the pipe diagram below, with a single packet in the pipe:



Q2.1 (1 point) What is the size of this packet?

- (●) xy
- ( ) $x + y$
- ( ) $xy + z$
- ( ) xz
- ( ) $x + z$
- ( ) $x + yz$
- ( ) yz
- ( ) $y + z$
- ( ) Not enough information

Solution:

x is how many seconds it takes to transmit the packet.

y is the bandwidth of the link, i.e. how many bytes can be sent per second.

Multiplying x (seconds) and y (bytes/second) gives the overall size of this packet.

Q2.2 (1 point) How long does it take to send a packet of the same size as the packet shown?

(Count from the time the first byte is sent, to the time the last byte is received.)

- ( ) xy
- ( ) $x + y$
- ( ) $xy + z$
- ( ) xz
- (●) $x + z$
- ( ) $x + yz$
- ( ) yz
- ( ) $y + z$
- ( ) Not enough information

Solution:

x is how many seconds it takes to transmit the packet.

z is the propagation delay.

Adding the transmission delay and the propagation delay gives the total transmission time for this packet.

Q2.3 (2 points) How long does it take to send 7 packets, all of the same size as the packet shown?

(Count from the time the first byte of the first packet is sent, to the time the last byte of the last packet is received.)

○ $7xy$	○ $7x + y$	○ $x + 7y$
○ $7xz$	● $7x + z$	○ $y + 7z$
○ $7yz$	○ $7y + z$	○ Not enough information

Solution:

x is how many seconds it takes to transmit the packet. Therefore, it takes $7x$ seconds to transmit all 7 packets.

z is the propagation delay.

Adding the transmission delay and the propagation delay gives the total transmission time for this packet.

Q2.4 (2 points) What is the maximum number of packets that could be in the process of being sent along this link, at any given moment?

Don't worry about fractions and rounding (e.g. assume x, y, and z are defined such that all answer choices are integers).

○ $\frac{y}{x}$	○ $\frac{xy}{z}$	○ $\frac{x+y}{z}$
○ $\frac{z}{y}$	○ $\frac{xz}{y}$	○ $\frac{x+z}{y}$
● $\frac{z}{x}$	○ $\frac{yz}{x}$	○ Not enough information

Solution:

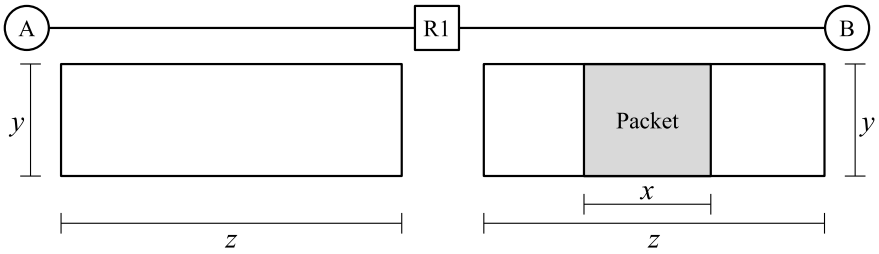
x is how many seconds it takes to transmit the packet. Therefore, it takes $7x$ seconds to transmit all 7 packets.

z is the propagation delay.

Adding the transmission delay and the propagation delay gives the total transmission time for this packet.

Now, consider the topology below. R1 has an infinite-size queue, and R1 continually processes packets in its queue in FIFO order. R1 can only start transmitting a packet once it has received the entire packet.

All packets in the question are the **same size** as the packet shown in the diagram below.



A wants to send a packet to B. No other packets are being sent along the links (i.e. suppose the packet in the diagram is not there).

In the next two subparts, select how long it takes to send the packet from A to B. Count from the time the first byte of the packet is sent at A, to the time the last byte of the packet is received at B.

Q2.5 (2 points) In this subpart, suppose R1 has **1 other packet** in its queue when A sends the packet.

How long does it take to send a single packet from A to B?

○ $x + z$	○ $y + 2z$	○ $2x + z$
○ $x + 2z$	○ $2y + 2z$	● $2x + 2z$
○ $2x + y + 2z$	○ $2x + 2y + 2z$	○ Not enough information

Solution:

Since all packets are the same size, by the time A transmits the packet onto the A-to-R1 link, R1 has finished clearing out its queue. Therefore, there will be no queuing delay at R1.

The packet takes $x + z$ time to travel from A to R1 (transmission delay x and propagation delay z). At time $x + z$, R1 has received the entire packet.

R1 then must send the packet along the R1-to-B link, which takes $x + z$ time again.

The total time to transmit the packet from A to B is $2(x + z) = 2x + 2z$.

Q2.6 (3 points) In this subpart, suppose R1 has **20 other packets** in its queue when A sends the packet.

How long does it take to send a single packet from A to B? Assume that the queue at R1 is not empty when this packet arrives at R1.

- ○ $20x + z$
- ● $21x + z$
- ○ $21x + 21z$
- ○ $20x + 2z$
- ○ $21x + 2z$
- ○ $20x + 20z$
- ○ $20x + 20y + 20z$
- ○ $21x + 21y + 21z$
- ○ Not enough information

Solution:

By the time the A-to-B packet arrives at R1, the queue is not empty (per the assumption). Therefore, we have to account for queuing delay.

More specifically, we have to wait for R1 to finish sending all 20 packets, and then it can send the A-to-B packet.

The time it takes for R1 to transmit 21 packets (the 20 in the queue, plus the A-to-B packet) is $21x$.

Then, the time it takes for the A-to-B packet to propagate along the R1-to-B link is z.

Thus, the total packet delay is $21x + z$.

Note that we don't need to consider the delay on the A-to-R1 link, because we already counted that time in the $21x$ term (i.e. the packet was sent from A to R1 while R1 was clearing out its queue).

Q2.7 (3 points) What is the maximum number of packets that can be queued at R1, such that when the last byte of the A-to-B packet reaches R1, there is no queue at R1?

Don't worry about fractions and rounding (e.g. assume x, y, and z are defined such that all answer choices are integers).

- ○ $\frac{x+y}{x}$
- ○ $\frac{xz}{y}$
- ● $\frac{x+z}{x}$
- ○ $\frac{x+z}{y}$
- ○ $\frac{xy}{z+x}$
- ○ $\frac{x+y+z}{x+z}$
- ○ xz
- ○ $x+z$
- ○ Not enough information

Solution:

The time it takes to send the packet from A to R1 is $x + z$.

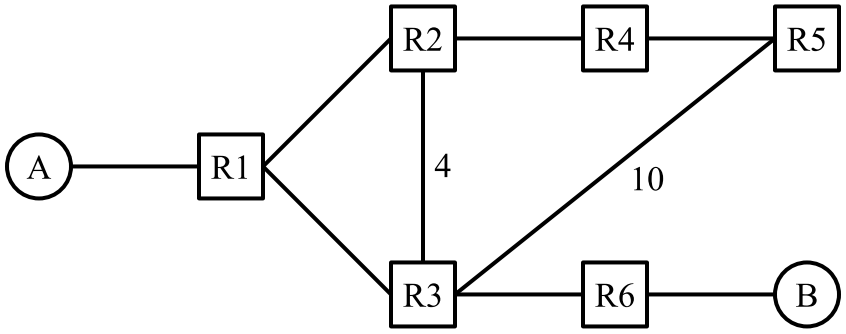
R1 needs x seconds to send out each packet.

Therefore, in $x + z$ seconds, R1 is able to send out $\frac{x+z}{x}$ packets.

Therefore, R1 can have at most $\frac{x+z}{x}$ packets in its queue, and all of those packets will be sent out by the time the last byte of the A-to-B packet reaches R1.

Q3 *Distance-Vector* **(12 points)**

Consider running the distance-vector protocol from lecture on the topology below. All unlabeled links cost 1.



At time $t = 0$, all forwarding tables are empty. At time $t = 1$, static routes are installed. At each subsequent time step, every router advertises all of its routes to all of its neighbors.

Assume that advertisements are sent, received, and processed on the same time step. For example, if R3 sends an advertisement at $t = 10$, then R1 receives the advertisement and updates its table entry at $t = 10$. R1 can then advertise this updated entry at $t = 11$.

Each subpart is independent unless otherwise stated.

Q3.1 (1 point) What is the first time step t where R5 will have a table entry for destination B?

$t = 3$

Solution:

At $t = 1$, R6 will have a table entry for destination B.

At $t = 2$, R3 will have a table entry for destination B.

Finally, at $t = 3$, routers R1, R2, and R5 will have a table entry for destination B.

Q3.2 (1 point) What is the first time step t where R5 will have a table entry with the least-cost route for destination B?

$t = 6$

Solution:

The least-cost route to destination B from router R5 is R5-R4-R2-R1-R3-R6-B. Therefore, it will take six time steps for R5 to record this path into its forwarding table.

Q3.3 (2 points) For this subpart only, split horizon and poison reverse are disabled.

A long time later, the network has converged. At this time, the R3-to-R6 link goes down.

R3 learns about this change and updates its table entry for B.

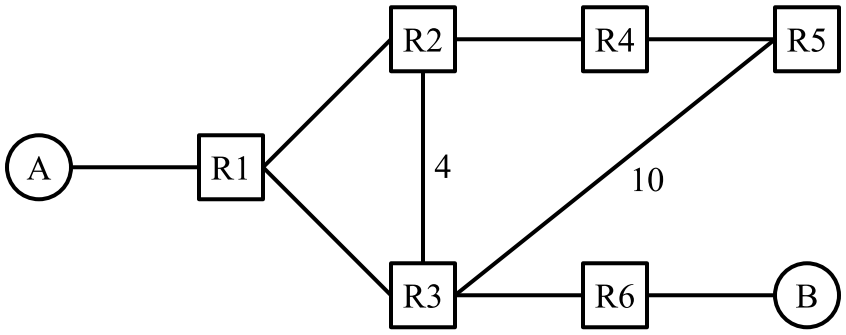
Which routers will R3 advertise this change to?

- ○ Only R1, because the change should only be advertised to routers using R3 as a next-hop.
- ○ Only R1, because R1 will propagate the change to the other routers eventually.
- ○ R1, R2, and R5, because the change should only be advertised to routers using R3 as a next-hop.
- ● R1, R2, and R5, because the change should be advertised to all neighbors.

Solution:

When a router changes one of the entries in its forwarding table, the distance-vector protocol says that the router must advertise this change to all of its neighbors.

The topology, reprinted for your convenience:



In the rest of the question, consider this scenario:

- Split horizon is enabled.
- No poison is ever sent.
- A long time later, the network has converged. After convergence, host A leaves the network.
- At time $t = 100$, R2 still has its entry for destination A (the same entry that it had at convergence).
- At time $t = 100$, all other routers have had their entries for destination A expired and deleted.

Q3.4 (6 points) At time $t = 103$, what table entry does each router have for destination A?

Note: $t = 103$ means you should consider three rounds of sending advertisements, receiving advertisements, and updating tables. The first round ($t = 101$) starts with R2 advertising its entry for A.

Some entries are filled in for you. For example, the bottom-right row says that R6's forwarding table has the entry "I can reach A with cost 7, via R3."

In the Cost column, write an integer, or ∞, or "N/A" (if the table has no entry for destination A).

In the Next-Hop column, write a router (e.g. "R1"), or "N/A" (if the entry has cost infinity, or if the table has no entry for destination A).

Router	Cost to A	Next-Hop to A
R1	**Solution:** 7	**Solution:** R3
R2	**Solution:** 8	**Solution:** R1
R3	**Solution:** 6	**Solution:** R2

Router	Cost to A	Next-Hop to A
R4	3	R2
R5	**Solution:** 4	**Solution:** R4
R6	7	R3

Solution:

See the next page for our full work.

At $t = 101$, R2 has an entry "I can reach A via R1." This is advertised to R3 and R4 (but not R1 because of split horizon).

At $t = 102$, R3 advertises to R1, R6, and R5 (but not R2 because of split horizon).

Also, at $t = 102$, R4 advertises to R5 (but not R2 because of split horizon).

At $t = 103$, R1 advertises to R2 (but not R3).

Q3.5 (2 points) Many time steps later, what happens to the routers' table entries for destination A?

(Reminder: All costs 16 or greater are infinite, and all costs 15 or less are finite.)

- ● Every router's entry has infinite cost.
- ○ Only R2 has an entry with infinite cost, and all other routers have an entry with finite cost.
- ○ R1, R2, R3 have an entry with infinite cost, and R4, R5, R6 have an entry with finite cost.
- ○ All routers have an entry with finite cost.

Solution:

See the next page for our full work (though you didn't need to run all these steps on the exam itself).

Intuitively, we've encountered the count-to-infinity problem. A nonexistent path is being advertised in the R1-R2-R3 loop, causing their costs to increase to infinity. Those high costs are also advertised to R4, R5, and R6, causing their entries to increase to infinity as well.

Solution:

The asterisk denotes an entry that updated at that time step. For example, at $t = 101$, R3 and R4 have their forwarding tables updated.

t = 100:
R2 can reach with cost 2 via R1 *

t = 101:
R2 can reach with cost 2 via R1
R3 can reach with cost 6 via R2 *
R4 can reach with cost 3 via R2 *

t = 102:
R1 can reach with cost 7 via R3 *
R2 can reach with cost 2 via R1
R3 can reach with cost 6 via R2
R4 can reach with cost 3 via R2
R5 can reach with cost 4 via R4 *
R6 can reach with cost 7 via R3 *

t = 103:
R1 can reach with cost 7 via R3
R2 can reach with cost 8 via R1 *
R3 can reach with cost 6 via R2
R4 can reach with cost 3 via R2
R5 can reach with cost 4 via R4
R6 can reach with cost 7 via R3

t = 104:
R1 can reach with cost 7 via R3
R2 can reach with cost 8 via R1
R3 can reach with cost 12 via R2 *
R4 can reach with cost 9 via R2 *
R5 can reach with cost 4 via R4
R6 can reach with cost 7 via R3

t = 105:
R1 can reach with cost 13 via R3 *
R2 can reach with cost 8 via R1
R3 can reach with cost 12 via R2
R4 can reach with cost 9 via R2
R5 can reach with cost 10 via R4 *
R6 can reach with cost 13 via R3 *

t = 106:
R1 can reach with cost 13 via R3
R2 can reach with cost 14 via R1 *
R3 can reach with cost 12 via R2
R4 can reach with cost 9 via R2
R5 can reach with cost 10 via R4
R6 can reach with cost 13 via R3

t = 107:
R1 can reach with cost 13 via R3
R2 can reach with cost 14 via R1
R3 can reach with cost 18 via R2 *
R4 can reach with cost 15 via R2 *
R5 can reach with cost 10 via R4
R6 can reach with cost 13 via R3

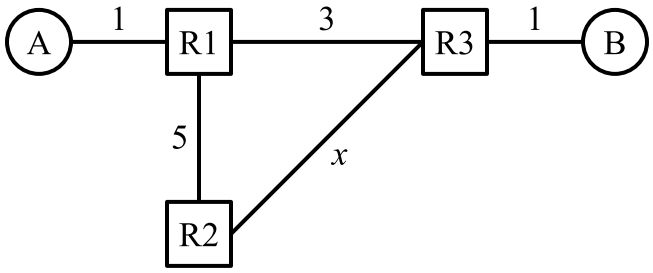
t = 108:
R1 can reach with cost 19 via R3 *
R2 can reach with cost 14 via R1
R3 can reach with cost 18 via R2
R4 can reach with cost 15 via R2
R5 can reach with cost 16 via R4 *
R6 can reach with cost 19 via R3 *

t = 109:
R1 can reach with cost 19 via R3
R2 can reach with cost 20 via R1 *
R3 can reach with cost 18 via R2
R4 can reach with cost 15 via R2
R5 can reach with cost 16 via R4
R6 can reach with cost 19 via R3

t = 110:
R1 can reach with cost 19 via R3
R2 can reach with cost 20 via R1
R3 can reach with cost 24 via R2 *
R4 can reach with cost 19 via R2 *
R5 can reach with cost 16 via R4
R6 can reach with cost 19 via R3

Q4 *Link-State* **(10 points)**

Consider the following network topology:



All routers are running the link-state protocol from lecture.

Each subpart is independent unless otherwise stated.

Q4.1 (2 points) In total, how many "hello" advertisements get sent between the routers? Do not count periodically re-sent advertisements.

6

Solution:

R1 sends a hello to R2, R3.

R2 sends a hello to R1, R3.

R3 sends a hello to R1, R2.

Note that hosts are not counted here, since the routing protocol is performed by the routers, not the hosts. Also, the question says to count advertisements sent between routers.

In the next two subparts, fill in the blank with a strict inequality (e.g. $x > 20$ or $x < 9$). Don't worry about cases where the result (loop or no loop) depends on tiebreaking between equal-cost routes.

Q4.2 (2 points) Assume **R1** doesn't know about the R1-to-R3 link. All other routers know the full topology.

If _____, then a packet sent from **A to B** would get stuck in a routing loop.

$x > 8$

Solution:

R1 is unaware of the R1-to-R3 link, so it forwards A-to-B packets to R2.

R2 knows the full topology.

If R2 forwards via R1, the cost to B is $5 + 3 + 1 = 9$.

If R2 forwards via R3, the cost to B is $x + 1$.

A loop forms if R2 decides to forward the packet back to R1. This occurs if the cost via R1 is less than the cost via R3. Writing this as an inequality, we get:

$9 < x + 1$

$x > 8$

Grading: Full credit was given for $x \geq 8$ since we said that you don't need to worry about the case of a tie (which here is $x = 8$).

Q4.3 (2 points) Assume **R3** doesn't know about the R1-to-R3 link. All other routers know the full topology.

If _____, then a packet sent from **B to A** would get stuck in a routing loop.

$x < 2$

Solution:

R3 is unaware of the R1-to-R3 link, so it forwards B-to-A packets to R2.

R2 knows the full topology.

If R2 forwards via R1, the cost to A is $5 + 1 = 6$.

If R2 forwards via R3, the cost to B is $x + 3 + 1 = x + 4$.

A loop forms if R2 decides to forward the packet back to R3. This occurs if the cost via R3 is less than the cost via R1. Writing this as an inequality, we get:

$x + 4 < 6$

$x < 2$

Grading: Full credit was given for $x \leq 2$ since we said that you don't need to worry about the case of a tie (which here is $x = 2$).

Q4.4 (1 point) What feature of the link-state protocol will help update the forwarding table to remove these routing loops?

- ○ Routers send poison advertisements.
- ○ Packets have a TTL field, and the packet is dropped when the TTL reaches 0.
- ● Advertisements are periodically re-sent.
- ○ None of the above. The routing loops in the above scenarios will stay forever.

Solution:

Option 1 is false, because poison advertisements are not used at all in the link-state protocol.

Option 2 is false, as the TTL field does not explicitly update the forwarding table.

Option 3 is correct. Eventually advertisements will be re-sent and all routers will learn the correct network topology.

Q4.5 (1 point) What is a possible cause of the scenario where R1 doesn't know about the R1-to-R3 link?

- [x] Advertisement(s) get dropped.
- [ ] User packets get dropped.
- [ ] R1 is not running a correct shortest-path algorithm.
- [ ] The scenario where R1 doesn't know about a link will never happen in the link-state protocol.

Solution:

The advertisement from R3 to R1 could've been dropped, which leads to R1 not knowing about its link to R3.

Option 2 is false, as user packets don't help routers learn about the network topology.

Option 3 is false, as the shortest-path algorithm is a separate step from learning about the network topology using advertisements.

Q4.6 (2 points) Suppose a routing loop has formed between R2 and R3.

R2 forwards a user packet to R3, and then the same packet is sent back to R2.

According to the link-state protocol from lecture, what will R2 do as a result of receiving this packet?

- [ ] Forward the packet to R3, and remove an entry from its forwarding table.
- [ ] Forward the packet to R3, and add an entry to its forwarding table.
- [ ] Forward the packet to a different router (not R3), and leave the forwarding table unchanged.
- [x] Forward the packet to R3, and leave the forwarding table unchanged.

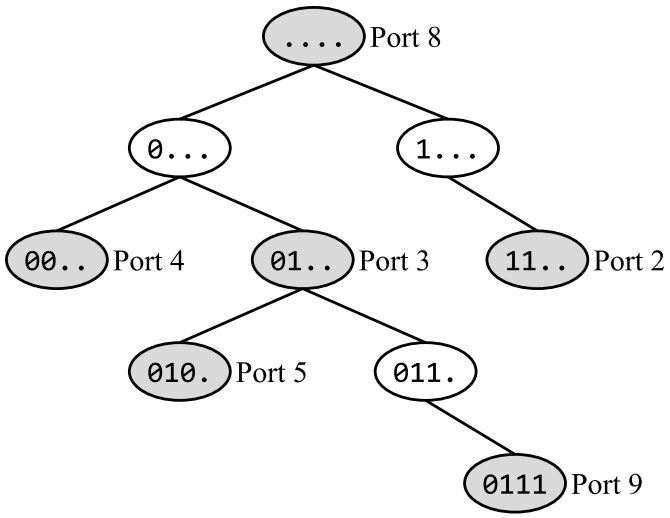
Solution:

Routers use destination-based forwarding. If R2 forwards a packet to R3 and then receives that packet again, R2 will still make the same forwarding decision, since the destination of that packet is still the same.

The link-state protocol only updates the forwarding tables based on routing advertisement packets. User packets are simply forwarded based on destination, and in the link-state protocol, user packets do not cause the forwarding table to change.

Q5 *Longest Prefix Matching* **(10 points)**

In this question, consider the longest-prefix-matching trie below. All subparts are independent.



Q5.1 (1 point) What part of the router is responsible for building this trie?

- ○ Data plane
- ● Control plane
- ○ Management plane

Q5.2 (1 point) True or false: The port numbers in this trie can be used for demultiplexing between different applications at the end host.

- ○ True
- ● False

Solution: Q5.1:

The control plane runs routing protocols and populates the forwarding table. The trie is a compact representation of that forwarding table.

Solution: Q5.2:

False. The port numbers in a trie refer to physical ports, the actual holes you plug cables into. They are not logical ports, which are used to associate packets with specific applications on a machine.

Q5.3 (2 points) How many of the 16 possible 4-bit IP prefixes are forwarded along the default route?

(Your answer should be a number between 0 and 16, inclusive.)

4

Solution:

All the 4-bit IP prefixes in the form `10..` will get forwarded along the default route. There are 4 such prefixes: `1000`, `1001`, `1010`, and `1011`.

The other 12 prefixes (with the form `00..` or `01..` or `11..`) will not be forwarded along the default route.

Q5.4 (2 points) In this subpart only, suppose we delete the node labeled `010.` (Port 5).

How many of the 16 possible 4-bit IP prefixes are forwarded along a different port using the modified trie (compared to the original trie)?

(Your answer should be a number between 0 and 16, inclusive.)

2

Solution:

The prefix `0100` was forwarded along Port 5, but is now forwarded along Port 3.

The prefix `0101` was forwarded along Port 5, but is now forwarded along Port 3.

All other forwarding decisions remain unchanged.

Q5.5 (2 points) Which of these IP prefixes represents exactly the set of IPv4 addresses that will get forwarded along Port 2?

○ `0.0.0.192/2`	○ `0.0.0.192/30`	○ `12.0.0.0/30`
● `192.0.0.0/2`	○ `192.0.0.0/14`	○ `12.255.255.255/28`
○ `192.0.0.0/30`	○ `12.0.0.0/2`	○ `12.255.255.255/30`

Solution:

IPv4 addresses are 32 bits, so the full prefix is:

```
11...... ........ ........ ........
```

Filling in zeros in all the unset bits:

```
11000000 00000000 00000000 00000000
```

Converting to dotted quad notation:

```
192.0.0.0
```

Finally, add the slash to indicate that two bits are fixed:

```
192.0.0.0/2
```

Q5.6 (2 points) How many 32-bit IPv4 addresses are forwarded along Port 3?

Reminder: $2^k - 2^{k-1} = 2^{k-1}$

○ 2^2	○ 2^{12}	○ 2^{24}	○ 2^{29}	○ 2^{31}
○ 2^4	○ 2^{16}	● 2^{28}	○ 2^{30}	○ 2^{32}

Solution:

Port 3 corresponds to the `01..` node. In this prefix, there are 30 bits unset, so there are 2^{30} IPv4 addresses in this range.

However, we have to subtract all the addresses in the `010.` range (29 bits unset, 2^{29} addresses), and the `0111` range (28 bits unset, 2^{28} addresses), since they get forwarded out of different ports.

This gives us a total of $2^{30} - 2^{29} - 2^{28} = 2^{28}$ addresses forwarded along Port 3.

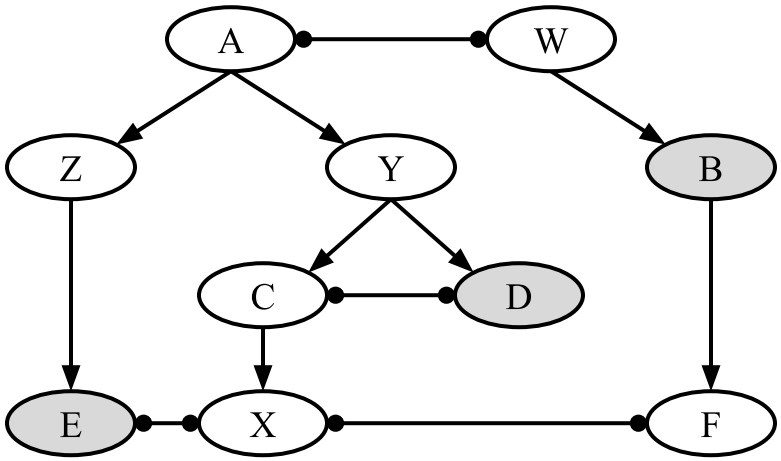
Another method is by realizing that only IPv4 addresses with `0110` as its prefix will be forwarded along Port 3, according to the trie. This means the next 28 bits can each be either 0 or 1, totalling to 2^{28} possible addresses.

Q6 *BGP: Shady ASes with Shady Policies* **(16 points)**

In the graph below, the shaded ASes (B, D, and E) use a special policy (instead of the Gao-Rexford policies):

- **Import policy**: Instead of Gao-Rexford, prefer the advertisement from the AS that comes first alphabetically. All subsequent tiebreaking rules remain unchanged.
- **Export policy**: When you receive a path, advertise it to all neighbors.

The non-shaded ASes use the standard Gao-Rexford import and export rules.



In the next three subparts, select whether the given AS path is valid (even if it is not necessarily the path used to forward packets). In other words, select whether all ASes along the path will advertise the path.

Q6.1 (1 point) X — E — Z — A — Y

● Valid ○ Invalid

Solution:

E agrees to participate (even though it's not making money), because it advertises paths to all neighbors.

Z agrees to participate, because neighbor E is a customer.

A agrees to participate, because neighbors Z and Y are both customers.

Q6.2 (1 point) X — C — D — Y

● Valid ○ Invalid

Solution:

C agrees to participate, because neighbor X is a customer.

D agrees to participate (even though it's not making money), because it advertises paths to all neighbors.

Q6.3 (1 point) Z — E — X — F

○ Valid ● Invalid

Solution:

X will not participate, as both neighbors are peers, and therefore X will not gain any money from advertising this path.

Q6.4 (2 points) What path will packets take from X to B?

○ X — F — B ○ X — C — Y — A — W — B

● X — E — Z — A — W — B ○ X — C — D — Y — A — W — B

Solution:

X — F — B is not a valid path, because AS F will not participate (no customer on either side).

The other three paths are valid. You can verify this by checking each intermediate AS and noting that all of them either agree to participate by the Gao-Rexford rules (if unshaded), or are shaded (which means they will always participate).

X is not a shaded AS, so it follows the standard Gao-Rexford import policies. X receives advertisements from E (peer) and C (provider), and prefers the path from the peer.

The path advertised by E (peer) is X — E — Z — A — W — B.

Q6.5 (2 points) For this subpart only, suppose X is a shaded AS. What path will packets take from X to B?

Assume that if an AS receives two paths, it will import the one preferred path, and only export that one preferred path.

○ X — F — B ○ X — C — Y — A — W — B

○ X — E — Z — A — W — B ● X — C — D — Y — A — W — B

Solution:

As in the previous subpart, X — F — B is not a valid path, and the other three paths are valid.

For this subpart, we are assuming that X is a shaded AS, so it follows the alphabetical import policy instead of the Gao-Rexford import policies.

X receives advertisements from E and C, and prefers the advertisements with next-hop C (alphabetically earlier), over the path with next-hop E (alphabetically later).

C receives two advertisements, from D and Y. Since C is not shaded, it follows the Gao-Rexford rules and prefers the path via D (peer) over the path via Y (provider). Per the assumption in the question, C only exports the preferred path via D, so the path that X hears about is X — C — D — Y — A — W — B.

Q6.6 (2 points) Are these special policies always profitable for any AS graph (not necessarily the one above)? In other words, do these special policies always help the ASes make money and save money?

○ Only the import policy is profitable. ○ Both policies are profitable.

○ Only the export policy is profitable. ● Neither policy is profitable.

Solution:

The import policy is not profitable. If you receive a path from customer Z and provider A, the profitable option is to import from customer Z, but the special policy instead imports from provider A (losing money).

The export policy is also not profitable. If you receive a path from a peer (doesn't make money), you should not advertise this path to a peer (you're participating in a path that doesn't make money). However, the special policy would advertise this path to the peer, causing you to participate in a path that doesn't make any money.

Q6.7 (1 point) For any AS graph with one or more shaded ASes (not necessarily the one above), are paths valley-free if the shaded ASes follow the special policies?

○ Always ● Sometimes ○ Never

Solution:

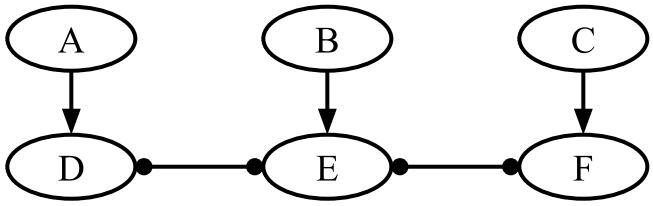
Some paths are valley-free. For example, consider X — C — Y — A — Z in the graph above.

Some paths are not valley-free. For example, consider a graph with P → Q ← R, where AS Q is shaded. Then the path P — Q — R has a valley, but Q agrees to participate in the path, and packets are forwarded along this path.

More generally, paths will still be valley-free if they don't involve any intermediate shaded ASes. In this case, all intermediate ASes follow the Gao-Rexford export rules, which ensures a valley-free path.

However, paths with intermediate shaded ASes may not be valley-free. The shaded AS might have providers on either side (forming a valley), but would still agree to advertise and participate in the path.

Q6.8 (3 points) In this subpart, consider this AS graph:



Select the minimum set of ASes that must be shaded in order for every pair of ASes to have a valid path between them.

(Reminder: A "valid" path is one where all ASes along the path will advertise that path.)

- ☐ A
- ☐ B
- ☐ C
- ■ D
- ■ E
- ■ F

Solution:

The only path between A and C is A — D — E — F — C. Along this path, D, E, F must be shaded so that they will participate in the path despite not making money.

Once these 3 ASes are shaded, you can check all the other pairs of ASes and notice that they all have valid paths between them.

In the next three subparts, complete the following statement:

To implement the special import policy, _____ (1) routers must set the _____ (2) route attribute when sending _____ (3) messages.

Q6.9 (1 point) Blank (1):

- ○ internal
- ● border
- ○ both internal and border

Q6.10 (1 point) Blank (2):

- ● LOCAL PREF
- ○ ASPATH
- ○ MED

Q6.11 (1 point) Blank (3):

- ● iBGP
- ○ eBGP
- ○ both iBGP and eBGP

Solution:

Recall that in regular BGP, the Gao-Rexford rules are encoded by having border routers set high LOCAL PREF values for customers and lower LOCAL PREF values for peers and providers. Then, the border routers send iBGP messages with the LOCAL PREF attribute set.

To implement the special import policy, we can use the same idea, but set the LOCAL PREF according to the alphabetical preference, instead of the Gao-Rexford preferences.

Q7 *Evil Traceroute* **(26 points)**

You are the operator of the Soda Hall router. (This means that all packets to and from Soda Hall are forwarded through your router.)

In this question, your goal is to trick any Soda Hall users who are running traceroute. If a Soda Hall end host is trying to run traceroute on the destination `4.4.4.4`, you should cause their implementation to return this fake path of 20 routers, followed by the destination:

```
["10.0.0.1", "10.0.0.2", "10.0.0.3", ... , "10.0.0.20", "4.4.4.4"]
```

Otherwise, you should process the packet as a router normally would.

Fill in `evil_traceroute` to implement this behavior. You can assume that all packets arriving at the Soda Hall router will be processed using this function. You can also assume that the traceroute users always probe using port 33434, and non-traceroute users never use port 33434.

```
1  # The fake path to 4.4.4.4 to show the user.
2  path = ["10.0.0.1", "10.0.0.2", "10.0.0.3", ... , "10.0.0.20"]
3
4  def evil_traceroute(raw_bytes):
5
6      # Parse the packet.
7      ip = IPv4(raw_bytes)
8      udp = UDP(raw_bytes[20:])
9
10     # If user is running traceroute, do evil things.
11     if (Q7.1 and Q7.2 and Q7.3):
12
13         if (Q7.4):
14             reply_packet = make_icmp_packet(src_ip    = Q7.5,
15                                             dst_ip    = Q7.6,
16                                             error_msg = Q7.7)
17         else:
18             reply_packet = make_icmp_packet(src_ip    = Q7.8,
19                                             dst_ip    = Q7.9,
20                                             error_msg = Q7.10)
21
22         send(reply_packet, next_hop = Q7.11)
23
24     # Otherwise, do what a router normally would.
25     else:
26         processed_bytes = process_packet(raw_bytes) # Q7.12
27         send(processed_bytes, next_hop = Q7.13)
```

Q7.1 (1 point) Which of these conditions helps you check if the packet is from a Soda Hall user running traceroute?

- ○ `ip.src == "4.4.4.4"`
- ○ `ip.src == "10.0.0.1"`
- ● `ip.dst == "4.4.4.4"`
- ○ `ip.dst == "10.0.0.1"`

Solution: The Soda Hall user running traceroute would be probing the destination `4.4.4.4`.

Q7.2 (1 point) Which of these conditions helps you check if the packet is from a Soda Hall user running traceroute?

- ○ `udp.src_port == 33434`
- ○ `udp.src_port == udp.dst_port`
- ● `udp.dst_port == 33434`
- ○ `udp.src_port != udp.dst_port`

Solution: The Soda Hall user running traceroute would be probing the destination using the traceroute port 33434.

Q7.3 (1 point) Which of these conditions helps you check if the packet is from a Soda Hall user running traceroute?

- ● Check if `ip.src` is in the range of addresses allocated to Soda Hall.
- ○ Check if `ip.dst` is in the range of addresses allocated to Soda Hall.
- ○ Check if the router's IP address is in the range of addresses allocated to Soda Hall.
- ○ Check if `4.4.4.4` is in the range of addresses allocated to Soda Hall.

Solution: If the packet is from a Soda Hall user, its source address must be in the range of IP addresses allocated to Soda Hall.

Q7.4 (2 points) What condition goes in the blank?

- ○ `if (ip.ttl > 0)`
- ● `if (ip.ttl > 20)`
- ○ `if (ip.ttl > 4)`
- ○ `if (ip.ttl > 30)`

Solution: If the user sends a probe with TTL greater than 20, then their probe would reach `4.4.4.4` along our fake path. Therefore, we should make a fake ICMP Port Unreachable packet and send it back to the user.

Otherwise, if the user sends a probe with TTL less than or equal to 20, then their probe would expire at one of the routers along our fake path. Therefore, we should make a fake ICMP TTL Exceeded packet and send it back to the user.

In the next three subparts, you'll fill in the fields of `reply_packet` for the first case. Your answers can be a hard-coded string/integer or a Python expression. Your answers can use any of the variables in the answer choices of the previous subparts, e.g. `ip.src` or `ip.ttl` or `udp.src_port`.

Q7.5 (2 points) Source IP:

`ip.dst` or 4.4.4.4

Solution: We're building an ICMP Port Unreachable packet, which should come from the destination `4.4.4.4`.

Q7.6 (1 point) Destination IP:

`ip.src`

Solution: The ICMP Port Unreachable packet should be returned to the Soda Hall user. The original probe packet came from the Soda Hall user, so their IP must be `ip.src`.

Q7.7 (1 point) ICMP Error Message:

● ICMP Port Unreachable ○ ICMP Time Exceeded

Solution: Remember, the goal is to send back a fake ICMP Port Unreachable packet in the case that the user sent a probe with a high enough TTL to reach the `4.4.4.4` destination.

Grading: Half-credit was given if you selected ICMP Time Exceeded. Some traceroute implementations will immediately terminate after seeing the destination IP address, regardless of what ICMP error is returned by that IP address. Sending back an ICMP Time Exceeded packet would cause these traceroute implementations to return the desired fake path of routers.

However, this alternate solution would not work on traceroute implementations that do check for the destination returning ICMP Port Unreachable, so it was not worth full credit.

In the next three subparts, you'll fill in the fields of `reply_packet` for the second case. Your answers can be a hard-coded string/integer or a Python expression.

Q7.8 (4 points) Source IP:

```
path[ip.ttl - 1]
```

Solution: We're building a fake ICMP TTL Exceeded packet, which should come from one of the routers on the path.

Which router sends the TTL Exceeded packet depends on the TTL in the user's traceroute probe.

If the user sent a probe with `TTL = 1`, the packet we send should look like it came from `10.0.0.1`, which is `path[0]`.

Q7.9 (1 point) Destination IP:

```
ip.src
```

Solution: Same as the previous case. The ICMP TTL Exceeded packet should be returned to the Soda Hall user. The original probe packet came from the Soda Hall user, so their IP must be `ip.src`.

Q7.10 (1 point) ICMP Error Message:

○ ICMP Port Unreachable ● ICMP Time Exceeded

Solution: Remember, the goal is to send back a fake ICMP TTL Exceeded packet in the case that the user sent a probe with a low enough TTL that it would hit one of the routers along the fake path.

In the next three subparts, `table` is the router's forwarding table (similar to the table in Project 2). You can ignore IP address aggregation in these subparts.

Q7.11 (1 point) What next-hop should we forward `reply_packet` to?

● `table[ip.src]` ○ `table[udp.src_port]`

○ `table[ip.dst]` ○ `table[udp.dst_port]`

Solution: The evil traceroute packets we create (faked ICMP packets) should be sent back to the user. The user's IP address is `ip.src`.

Q7.12 (3 points) What processing does the router need to do on a non-traceroute packet before sending it out? Select all that apply.

- ☐ Increase the TTL by 1.
- ■ Decrease the TTL by 1.
- ■ Recompute the IPv4 checksum.
- ☐ Recompute the UDP checksum.
- ■ Fragment the packet if it's too big.
- ☐ Unwrap the IP header to process the UDP packet.
- ○ None of the above

Solution: Note that we don't touch the UDP header or recompute the UDP checksum, because Layer 4 behavior is implemented only at end hosts, not at routers.

Q7.13 (1 point) What next-hop should we forward non-traceroute packets to?

- ○ `table[ip.src]`
- ○ `table[udp.src_port]`
- ● `table[ip.dst]`
- ○ `table[udp.dst_port]`

Solution: For non-traceroute packets, we should just do standard destination-based forwarding. The destination of the packet is `ip.dst`.

Q7.14 (1 point) Where would the `evil_traceroute` function most likely be implemented?

- ● Data plane
- ○ Control plane
- ○ Management plane

Solution: Forwarding packets is handled by the data plane.

Control plane is not the best answer here, because we aren't running any routing protocols, and we aren't communicating with other routers.

Management plane is not the best answer here, because we aren't reporting any errors to the operator, and the operator isn't sending any configuration data to the router.

In the next two subparts, complete the following statement:

A router implementing `evil_traceroute` would most likely handle traceroute packets as _____(1), and handle other packets as _____(2).

Q7.15 (1 point) Blank (1):

○ user packets ● punt traffic ○ control traffic

Q7.16 (1 point) Blank (2):

● user packets ○ punt traffic ○ control traffic

Solution: Traceroute packets need special processing, so they'll likely be implemented as punt traffic (using the slower path and reaching the controller card).

Other packets can just be directly forwarded, so they'll likely be sent as user packets (along the fast path).

Q7.17 (1 point) Would the `evil_traceroute` function work as intended if some IPv4 packets had additional Options in their headers?

○ Yes, `evil_traceroute` would just ignore the Options.

○ Yes, `evil_traceroute` would correctly process the Options.

● No, because of Line 8: `udp = UDP(raw_bytes[20:])`

○ No, because of Line 27: `send(processed_bytes, ...)`

Solution: If the IPv4 packet has additional Options, then the IPv4 header is no longer 20 bytes long. As a result, Line 7 will not correctly extract the UDP header anymore. The hard-coded number 20 is assuming that the IPv4 header has no options and is 20 bytes long.

In the last two subparts, consider a user in Soda Hall running traceroute. The user always sends traceroute packets from Port 29130, and always probes using Port 33434.

The user sends out a traceroute packet, and receives a response packet from the router running `evil_traceroute`.

Q7.18 (1 point) If the response packet contains a UDP Source Port header field, what is the value of this header field?

- (●) 29130
- ( ) 33434
- ( ) The response does not contain a UDP Source Port field.

Solution: The intended answer is 29130. Recall that ICMP-over-IP packets contain the first few bytes of the original packet that the user sent. The user's original packet had a source port of 29130.

Grading: We accepted "The response does not contain a UDP Source Port field." This is not the correct behavior of ICMP as shown in the project. However, the provided code seems to be calling `make_icmp_packet` without passing in the `raw_bytes` of the packet as an argument. Since the code looks misleading, we gave students the benefit of the doubt and accepted this alternate answer.

Q7.19 (1 point) If the response packet contains a UDP Destination Port header field, what is the value of this header field?

- ( ) 29130
- (●) 33434
- ( ) The response does not contain a UDP Destination Port field.

Solution: The intended answer is 33434. See solution to previous subpart. The user's original packet had a destination port of 33434.

Grading: We accepted "The response does not contain a UDP Destination Port field." This is not the correct behavior of ICMP as shown in the project. However, the provided code seems to be calling `make_icmp_packet` without passing in the `raw_bytes` of the packet as an argument. Since the code looks misleading, we gave students the benefit of the doubt and accepted this alternate answer.

Comment Box

Congrats for making it to the end of the exam! Leave any thoughts, comments, feedback, or doodles here.

Nothing in the comment box will affect your grade.

Ambiguities

If you feel like there was an ambiguity on the exam, you can put it in the box below.

For ambiguities, you must qualify your answer and provide an answer for both interpretations. For example, "if the question is asking about A, then my answer is X, but if the question is asking about B, then my answer is Y." You will only receive credit if it is a genuine ambiguity and both of your answers are correct. We will only look at this box if you request a regrade.