Question: A-1 Motivate why the lecture notes define anonymity roughly as “the IP is unknown”.
Answer: The lecture notes take the position that if the IP address is disclosed, anonymity is broken. As long as the IP is kept hidden, we maintain anonymity. This is due to the fact that the provider of the address (usually the ISP) is not to be trusted, and in a security context we should be able to chose who to trust ourselves. Hence, if the IP is revealed, it is assumed that it can be connected to the actual user behind it. This can be achieved either by malicious acts or by action from law enforcement and is therefore as reasonable definition.

Question: A-5 What is the purpose of the random value R_0 in a Mix?
Answer: R_0 is a random value that is added to the message to avoid simple brute force guessing attacks. If we don’t have R0, the message that would leave the mix would be just (K_a(M), A) and since the public key (K_a) for that person is publicly known to everyone, it would be easy to use it on a large number of messages (e.g. M’) and see if they match, hence being able to “derive” the original. I.e. K_a(M’) =? K_a(M).

Question: A-6 What is the purpose of the random values R_1 in a Mix?
Answer: Similar to the question A-5 above, the random value R_1 is instead added to the receiver address part. However, R_1 here functions as a protection against being able to connect the input and output messages sent to/from of the Mix. If not used, the message going into the mix would just be (K_1(K_a(R_0, M),A)) and then by looking at the output (K_a(R_0,M),A) and encrypting it with the Mix’s public key, we could easily see which input message it corresponds to.

Question: A-9 Describe the strongest adversary possible and explain why two fundamentally different anonymizing designs have intrinsically different possibilities of protecting against such an adversary.
Answer: The strongest adversary possible is often referred to as the Global Passive Adversary (GPA). The GPA can see all traffic everywhere on the network and eavesdrop on all sides of a node. Thus, it also knows the senders’ and receivers’ addresses. The GPA can use traffic analysis methods to try to determine who communicates with whom, and when. The two anonymizing designs referred to in the question are;

A) *High-latency designs:* These are suited for emails, etc. that are not time critical. By using mix-nets for example, “enough” data can be encrypted/protected, mixed and batched together, protecting well against the GPA who couldn’t figure out what goes to whom.
B) *Low-latency designs:* These are for (close to) real time traffic like browsing the web, etc. The time factor makes it hard to obfuscate the communication using e.g. mix-nets as small batch sizes are less safe. I.e. the anonymity set becomes very small and one could could look at e.g. timing of incoming/outgoing traffic to analyze the connection between a sender and receiver.

Question: A-11 When returning a message using an untraceable return address, why does each Mix encrypt the return message with the randomness R_i?

Answer: By encrypting the return message with the random value R_1, you accomplish two things; 1) similar to the randomness described in questions A-5 and A-6, it prevents an adversary from using the Mix's public key to encrypt other addresses and figure out the right one, and 2) prevent eavesdropping to be used to simply compare messages entering/leaving the mix. I.e. if not used, the same message ($K_x(R_0,M)$) would be used on both sides of the Mix.

Question: A-12 Regarding replay attacks on Mixes, two protections are suggested in the lecture notes. Which? Would you say that any of them is the better choice? Show how the two strategies can be combined and how this can make the protection more efficient.
Answer: Using a so called replay attack, an old message is sent ending up at the exact same place and by comparing the messages from our two different batches we can identify the receiver. Just like a "good" Mix doesn't allow duplicate messages to be processed (removing one of them for the same conceptual reason), the Mix could save e.g. a hash of every and each of all the messages it processes (during a practical time frame). It can then compare all incoming messages (hashed) with the historical database and remove "replayed" ones. Another countermeasure is to use a timestamp to make sure it is a fresh message coming through, else discard it. It would be essential to protect the timestamp from tampering obviously.

I would personally go with the timestamp-solution if I had to chose between the two, as it does not require persistent and safe storage and resources to maintain the historical database (which also would need protection). However, it does come at the cost of a proper design and keeping a "secure" internal system clock. If the two options were combined, the practical implementation could be made more efficient by just storing (remembering) recent messages in the database, then discarding them after certain time threshold (as we know the time stamp) making it more reliable on small time discrepancies.

Question: A-14 Consider the long term intersection attack. Explain how the number of users and the sizes of each batch will affect the efficiency of the attack.
Answer: The long term intersection attack works even if the Mixes are perfect as long as the adversary can monitor all incoming and outgoing messages and have a good guess on when entered messages are leaving. Further assume that multiple messages can be linked back to the same sender as e.g. in the case of multiple posts under a pseudonym on a message board.If you store all the senders over time and intersect these sets, you will have the "right" sender left at last.

Increasing the traffic (from all potential users users) is the main way to protect against the attack. This can be done via dummy traffic, making all users look active all the time, and the adversary will then have a harder time to link messages to users. The problem comes from the initial anonymity set, that often decreases over time rather than depending on batch size or number of actual users.

Question: A-21 In August 2013, a large botnet used Tor Hidden Services to communicate with the Command and Control server. This resulted in an increase from 1 million to 6

million daily clients using Tor in just three weeks. As a result, the time required for downloading a 50 KiB file doubled from 1.5 seconds to 3.0 seconds. However, the amount of traffic on the network did not change very much? Why is that?
Answer: The problem was likely that the it took longer to find suitable nodes to use in establishing the circuit, but once they were found, the traffic flowed just like normal via them. The traffic flowing to/from the HS/C&C could also have been extremely sparse, just a few simple commands once in awhile. So latency, not bandwidth, would have been the issue. (Also, from where were the files downloaded and to where?)