Question 1:
A block cypher with an 8 bit block size is very easy to break with a known-plaintext attack. Describe how you would do so. Hint, there's a version of this exact challenge in many newspapers daily (They're called cryptograms, here's an example).

Answer 1:
If you have plain text for an 8 bits you can just compare the cyphertext to see what each corresponding char represents in cyphertext. If you can compare those it should be simple to decode. If you didn't have the plaintext as in the cryptogram, you can look for the smallest and most common words to start. Examples (a, the, are), and use the decoded letters from those to get clues to the rest of the words.

Question 2:
Assume you're sending a long message using a block cypher (like AES) with the following scheme: split the message into blocksize chunks, then encrypt each with the same key. Basically Alice sends Bob AES(m1, k), AES(m2, k), AES(m3, k), etc.

- **a(3 points): Even if they can't decrypt blocks, what information can an eavesdropper discern from this scheme? Hint: imagine Alice is sending a table of data with fixed-length rows**

In this example, they are reusing the same key every time. So every time something is encrypted, it is encrypted in the same way. That means that an eaves dropper would be able to see patterns. If there are two cyphertext blocks that are the same, it is likely that the plaintext of both those blocks are the same. There is leaked information. That is part of the reason in a real world application they use a different key for each round.

- **b(4 points): Things are actually even worse! A malicious attacker can actually CHANGE the message that Bob receives from Alice (slightly). How? This is particularly bad if the attacker knows the structure of the data being sent (like in part a)**

  If an attacker modifies a block cypher it should mess up the entire block that was modified. This can render the resulting deciphered plain text unusable.

- **c(3 points): How could you modify the scheme to mitigate/prevent these types of attack?**

Change the key for every round. AES uses a new key for every round. Also send HMAC along with the message. HMAC = hashfunction that take message and key, runs twice.

HMAC(message, key) = H(key + H(key + message))