1. For task 1, how hard would it be for an adversary to solve the Diffie Hellman Problem (DHP) given these parameters? What strategy might the adversary take?
Given large prime modulus and generator, it would be infeasible to solve the Diffie-Hellman Problem with brute force methods since there are a large number of potential values for g and p. However, if the attacker has some knowledge about the system or can guess certain properties of g and/or p, they could potentially make progress towards solving the problem. They could try an exhaustive search of different privake keys determined by prime modulus, but given the size, this could be hard.

2. For task 1, would the same strategy used for the tiny parameters work for the large values of q and alpha? Why or why not?
In terms of trying to solve it, having tiny parameters would make it significantly easier to solve because the larger the prime modulus and generator is, the key space increases exponentially making it harder to brute force/try exhaustive search strategies.
In terms of trying to implement the strategy, this would also work for large parameters as well because using pow and directly converting hex to ints, this allows for our program to handle large numbers.

3. For task 2, why were these attacks possible? What is necessary to prevent it?
This attack is possible because these values have special properties. These values make it so that there is variability in the public keys, making the key exchange less secure. To prevent this, one must avoid special values such as the one's used in this task. One must pick a primitive root modulo q that don't include these special values.

4. For task 3 part 1, while it’s very common for many people to use the same value for e in their key (common values are 3, 7, 216+1), it is very bad if two people use the same RSA modulus n. Briefly describe why this is, and what the ramifications are.
If two users have the same modulus n that means that would essentially share the same public key. If an attacker gets the ciphertext for one person, they could use the same public key for the other user. This would allow them to decrypt messages from both users at once which would lead to serious security vulnerabilities.