Question 1:

To do question one, I went to the last else statement in playerenemycollision and removed the scheduled death. I added a variable for the health, then an if to check that the player's health was not null. Next, I went to the enemy controller and added a public variable for damage, setting it to 2 for all enemies in the inspector. Then, back in the else statement, I put playerHealth.Decrement(enemy.damage) so it would use the enemydamage int. Lastly, I went into the health script and changed the 1 in Decrement to the damage amount.

Question 2:

For question two, I went into the Health script and created a function called Immunity which starts a coroutine. The coroutine first changes the bool isImmune to true and writes a debug message for "is immune." Then I used a yield return new WaitForSeconds(amount), with amount currently set to one second (but it could be changed in the function call). Once the amount of seconds is up, isImmune is set to false. Then, back in the last if statement I was working on for question one, along with the null check for playerHealth, I added a check for !playerHealth.isImmune. If the player is immune, no damage will be dealt; otherwise, damage will go through and it will call playerHealth.Immunity(1).

Question 3:

For this question, I started with brute force. I made an object for the gun, with a fire point child and a prefab for the bullet. Then I went into PlayerController and made public references for the gun, float bulletSpeed, bulletPrefab, and a bool for flipped. Along with the movement logic, I added a GetButtonDown for Fire1, which, when activated, instantiated the bullet at the firePoint and then added a force forward at bulletSpeed. Once this worked, I moved on to making the damage work. I made a new class for bulletenemycollision. This took the enemyController and bullet GameObject. I made a variable to hold the enemyHealth, checked that bullet and enemyHealth were not null, then decremented enemyHealth by one and destroyed the bullet. Once this was working as I intended, I created a new class called PlayerShoot scheduled when the player pressed Fire1. I moved the original logic into this class, adding a bool check and reversing the bulletSpeed when the player is facing left. I also flipped the gun GameObject when the player faced left in the playerController script.