QUESTION 1

- **Preparation Phase:**

  - First, I create two scenes, Boot and GameScene. In my Boot scene, there will be Manager files (GameManager, UIManager, InputManager etc.), which will be Singletons. Game items will be included in GameScene. I prefer to use a single game scene even if there will be more than one level in the game. Because, instead of removing the current scene and loading a new scene, loading the game scene once makes the transition between levels much faster.

  - When I make a 2D platform game, I first start by creating the game scene and when necessary, I create Manager scripts and other scripts. In the game scene, I first start by creating Tilemap. First, I create the required Rule Override Tiles by importing the Tile assets. If the map to be drawn has already been given by the level designer, all that remains is to simply draw the desired level or levels.

  - In the next step, I put the Player object on the scene and if the animation assets are ready, I create the animator for the Player object and determine the necessary variables for animation transitions.

  - In the next step, I create scripts for Singleton.cs, Player.cs, PlayerAnimation.cs and InputManager.cs as first thing. Since I make games mainly for mobile, I have to get input with both the keyboard and the virtual buttons that I will place on the screen, and even I may need different input methods in the future, I will definitely create the InputManager script at the beginning. Since I will make both the InputManager and the other Manager files Singleton, I also create the Singleton.cs class that all of them will inherit. These classes will not take time since the Singleton class is already ready-made and I will only get keyboard input in the first step in the InputManager. In the first step in the PlayerAnimation class, I only write methods about Idle and Move animations. In the Player class, I write and test the Movement method that will provide the required movement according to the value of the move variable in the InputManager. If there is no problem in the test, the preparation phase is complete.

  - This process doesn't take too long as it's a repetitive phase for almost every game.

- ## Development Phase:

    - During the development phase, I first develop the Player. If player and other objects (Enemy etc.) have same functionalities, I create the abstract classes required for these common functions. If possible, I complete the player and start making adjustments for other objects. I create the necessary UI elements in a UIManager object and set the changes to be made in the UI according to the changing game states in the UIManager.cs script. The development phase is the phase that will cover most of the process (more than 90%).

- ## Test Phase:

    - I will test if all functions work on different devices and fix it if I see a problem. When I have completed all the problems, I make the necessary preparations for the target platform (such as application signing settings for Android) and get the output ready to be published. Although this process can take a long time depending on the problems that will arise during the testing phase, it is very short compared to the development phase.

QUESTION 2

- **2.a:**
    - Creating two screens as Boot and GameScene, creating the GameManager object on the Boot screen and writing the screen transition codes required for debug to the attached GameManager.cs file.
    - Creating Tilemap and giving a collider to Ground Tilemap
    - Creating player, setting the collider of the player and placing the player object in the game scene
    - Creating basic animations for the player
    - Writing Player.cs, PlayerAnimation.cs and InputManager.cs scripts and adding necessary movement codes. Attaching player scripts to Player object and InputManager script to GameManager object like all Manager scripts.

- **2.b:**
    -
    - What is the number of levels? How are the maps of the levels to create a Tilemap?
    - What functionalities will the player have? Will different players have specific functionalities? Will there be common functionalities that the player shares with other game objects?
    - What will be the frame rate for the animation assets given for the player and how will be the transitions between animations?
    - What kinds of input methods will there be in the game? Which platforms will the game run on?

QUESTION 4

- **4.a:**

    As I mentioned before, I create a class named InputManager and make the input decoupling there. I make the adjustments related to the UI in the UIManager class.