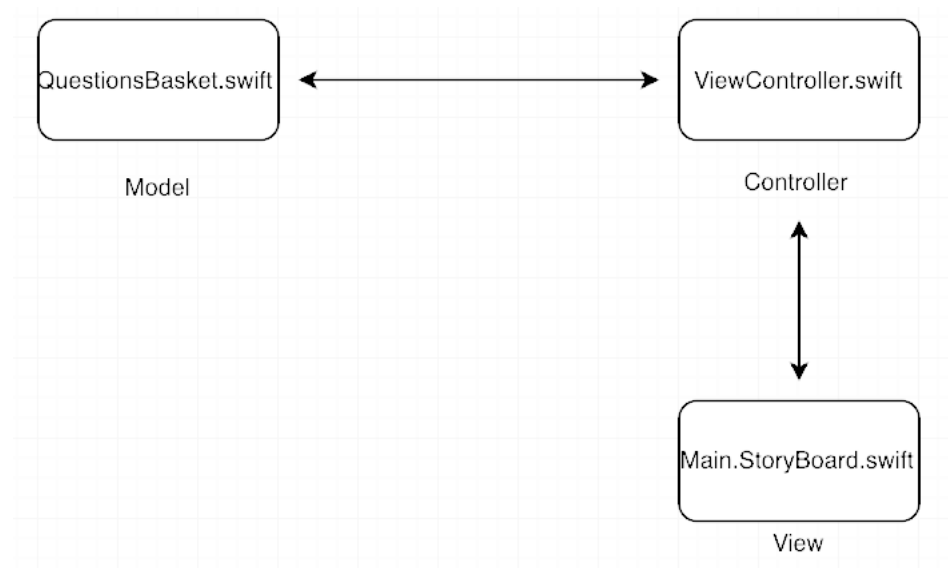Question 1)

The code has been re-factored to use MVC pattern.
It has folders Model , View, Controller and files are distributed into appropriate folders.
With the test codes.
https://github.com/chldbwnstm/cdma04C

Question 2)


```
┌──────────────────────┐                    ┌──────────────────────┐
│ QuestionsBasket.swift│ ◄───────────────► │ ViewController.swift │
└──────────────────────┘                    └──────────────────────┘
          Model                                    Controller
                                                        ▲
                                                        │
                                                        ▼
                                             ┌──────────────────────┐
                                             │ Main.StoryBoard.swift│
                                             └──────────────────────┘
                                                       View
```

Question 3)

As the video from lynda suggests, Xcode provides two testing frameworks which are unit testing and UI testing. Creating unit tests for at least the success case and failure case for each module in the program is best practice in Test-Driven-Development (TDD). Creating automated unit tests can be time consuming and costly. However, it is an investment for a larger return as it improves the quality of the code and saves you from multiple manual tests that you would need to do later.

For unit testing of my app, I have implemented the IOS unit testing module into my project. In the unit test, I have tested the only business logic in my app, which is testing if the chosen answer for a question is correct or not. Because the data type of answer is an Int, I have tested two cases which are: 1) when the answer is correct 2) when the answer is incorrect. This test assures both success and failure cases for the business logic which covers all the outcomes of the logic.

For UI testing of my app, I have implemented Xcode's UI testing framework. This was achieved by adding the IOS UI testing bundle to my project.
Then Xcode created a folder called 'my project name' followed by UITests.swift.
After that, I created a function called testQuizApp inside the file then clicked record button on the bottom of Xcode. Magically, the app launched by itself and then recorded all events into the function. After stopping the app, then when the test start button was clicked, all the UI recorded functions repeated itself and console showed that all tests were successful. The purpose of UI testing is that when there is some changes to the UI coding, the UI test will fail which gives the developer a notification that the UI has changed. This could help in debugging as the developer could had not wanted the UI to change and it was done by mistake. Also, it tests the first page – second page segue if the second page label exists or not.