iOS Developer Assessment

This is a technical assessment for an iOS developer using Swift as the programming language. The test will cover topics related to iOS app development, Swift language, and commonly used frameworks.

Exercise 1: Swift Basics
Topic: Swift Language

Task:
Write a function in Swift that takes an array of integers as input and returns the sum of all even numbers in the array.

```
func totalEvenNumbers(in arrayNumbers: [Int]) -> Int {
    var total = 0
    for item in arrayNumbers {
        if item % 2 == 0 {
            total += item
        }
    }
    return total
}
```

Exercise 2: Table View Integration
Topic: iOS App Development

Task:
Create an iOS app with a UITableView that displays a list of products. The app should fetch the product data from an API endpoint and display the product name and price in each table view cell.

Exercise 3: Core Data Integration
Topic: iOS App Development

Task:
Modify the previous app to use Core Data for local persistence. Add the ability to save the fetched product data to the Core Data database and load it from the database on app launch.

Exercise 4: MapKit Integration
Topic: iOS App Development

Task:
Create an iOS app that integrates MapKit and displays a map with annotations for a list of locations. The app should fetch the location data from an API endpoint and display pins on the map for each location.

Exercise 5: Unit Testing
Topic: iOS App Development

Task:
Write unit tests for the `sumOfEvenNumbers` function from Exercise 1 to ensure it works as expected for different input arrays.

Exercise 6: View Controller Lifecycle
Topic: iOS App Development

Task:
Explain the iOS view controller lifecycle and list the key methods that are called during different stages of the lifecycle.

viewDidLoad
This method is invoked only once upon creating the view and loading it into memory. However, it's worth noting that the view's bounds are not defined at this point. Typically, we override this method to perform initializations for objects that the view controller will utilize. It's important to remember to include a call to the 'super' method when overriding it.

viewWillAppear
This method is invoked just before the view becomes visible on the screen. When this method is called, the view's bounds are established, but the orientation hasn't been configured yet. It's important to exercise caution when working with appearance methods because they are triggered each time the view is displayed. Equally important, remember to include a call to the 'super' method

viewDidAppear
This method is called right after the view is visible to the user. It’s a good place to start animations. Same as before, we shouldn’t forget to call super.

viewWillDisappear
This is called when the view is about to disappear from the screen. One thing we can do here is to save the user data to not lose anything important. Another thing we might be canceling is network requests. We can override this method also if we want to hand over first responder jobs to another view when the view disappears.

viewDidDisappear
This is called when the view is disappeared from the screen. The view is removed from the view hierarchy at the moment.