Final Assessment (25%)

Instructions:

- Your code will be tested on an **iPhone 11** emulator. You are responsible for testing that your application runs and appears correctly on a simulator/device of the same configuration.

- Grading is based on the technical correctness of your implementation for the requested features. Besides implementing the required functionality, you must use the correct coding conventions demonstrated in class, professional organization of the code, alignment, clarity of function and variable naming, etc. Comments for code snippets are recommended but not required. Your application's UI does **not** need to be pretty. However, the UI elements must be clearly organized, legible, and easy to understand. Marks are deducted if the UI is messy, disorganized, or difficult to read.

XCode Project Requirements

- **You must name your XCode project as follows: FinalTest_FirstName,** example: **FinalTest_Peter**

- Your **AppDelegate.swift** file must include the debug code that outputs **the *path*** to where your emulator/device stores i**ts CoreData sqlite file**. An example of this code is provided in the CoreData lecture materials.

Submission Checklist:

- When you are done, create the zip of your entire project folder.
- Rename the zip file **FinalTest_*YourFirstName*.zip**, for example: ***FinalTest_Peter.zip***
- Upload the .zip file in the assessment dropbox.

Academic Integrity:

This is an individual assessment, which means you are to complete the work by yourself, without assistance from others. Others = other learners, friends outside the college, tutors, etc.

- Permitted activities:
  - Using the Internet to search for Swift/IOS syntax
  - Using the course materials
- Not permitted:
  - Sharing, discussing or distributing links, materials, full/partial solutions
  - Using full/partial solutions/tutorials found on the Internet
  - Using a "reference" or providing a "reference" to other learners
  - Asking others (both inside/outside the class) to help debug your code, or to code some/all of your application
  - Usage of homework help sites, like CourseHero.com, Chegg, or similar is a violation the college's copyright policy AND a breach of academic integrity

Task Description:

In this assessment, you will build an application that displays information about world countries. Your app must consist of the following screens:

1. Main Screen: This screen displays a list of countries
2. Country Details Screen: This screen displays information about a specific country.
3. Favourite List Screen: This screen displays a list of countries that the user favorited

Details about each screen are provided below. The user interface for this application must be built with Storyboards (not SwiftUI or XIBs).

Tab Bar Navigation

- The two screens of the application must be presented in a **Tab Bar Controller** (UITabBarController). Each tab must be labelled and an appropriate icon provided. If you are unfamiliar with using a TabBarController, you are permitted to research how to use it.

Screen #1: Get a Random Quote

- This screen displays a list of countries in a TableView. Use this API to fetch a list of countries: https://restcountries.com/v2/all
  - You are ***not*** required to use a APIFetcher singleton / pub/sub service layer pattern.
- Each row of the TableView must display the country name.
- Tapping on a row in the TableView will navigate the user to the **Country Details Screen**. Navigation to the Country Details Screen must be performed using **Navigation Controllers** and the storyboard?.instantiateViewController(:...) pattern.
- Screen #1 must *send* the country name, country code (alpha3Code), capital, and population to the **Country Detail Screen** as an object. It is possible for Screen #1 to send an nil object.

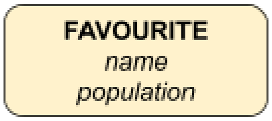
Screen #2: Country Detail Screen

- This screen receives the country object from Screen #1, and displays the country name, capital, code and population in UILabel(s). You must clearly mark the information, example:

  Country Name: Afghanistan
  Capital: Kabul
  Country Code: AFG
  Population: 27657145

- If the screen receives a nil object, the screen should display a message that says: "Sorry, no country information found" in a UILabel(s). When working with potentially nil objects, you must use either the guard let or if-let constructs.

- The screen must also contain a button called "ADD TO FAVORITES". Tapping on the button will *persist* the country's name and population to CoreData. A country can only be persisted once.
    - An alert box (popup) should be displayed indicating whether the country was successfully added (or cannot be added + the reason why it cannot be added).
    - The CoreData entity must conform to the following description:



- At any time, the user must be able to return to Screen #1 via the Navigation Controller's "Back" button.

Screen #3: Favourite List Screen

- Upon opening this screen, the app should fetch the user's favorite countries from Core Data and display the country name and population in a tableview. Use the built-in ***Subtitle*** table cell style.
- If the country has a population that is *larger* than Canada, set the background color of the row to a color of your choice.
    - Please use a color that is visually obvious / distinct from the default table row background color (example: yellow)
    - Please ensure that the *text* of the row is still legible on your selected background color
- The user can delete a favorite by swiping left on the table row. Deleting the favorite must remove the item from CoreData and the tableview.

END OF ASSESSMENT