Storyboards and Navigation

As today exercise we can design an app with multiple view controllers and complex navigation.

Let's imagine we want to create an utility app to solve different types of math problems. The user can navigate the problems using a **UITabBarController** to switch between them. Each individual panel in the tabBar is a **UINavigationController** each with two individual view controller, the first one has the UI components that allow the user to insert the necessary input data but the solution will be displayed on the second controller, which is pushed by triggering a "Solve!" button. Some input validation is always required, depending on the specific problem (i.e. empty fields, negative numbers, zeroes, etc) before navigating to the next controller: you can either perform the validation before you trigger the segue and then trigger the segue manually only after validation succeeds, using *performSegue(withIdentifier:sender:)* or alternatively trigger the segue every time the button is pressed but provide the delegate method *shouldPerformSegue(withIdentifier:sender:)* to prevent the segue to occur if validation fails. Optionally you can use the **UIAlertController** to display a simple error message when validation fails. The next view controller then will accept the input data and use it to solve the problem and somehow present the results.

Examples of problems to be solved included in your tabs, in rough order of complexity:

1. one tab can be used to solve second degree equations, by providing coefficients a, b, and c

2. one tab can be used to find the length of the third side of a right triangle by providing the other two sides, using the Pythagorean Theorem (optionally, the view controller containing the solution might also try to draw the triangle adding a triangular **CAShapeLayer** to any view)

3. one tab can be used to print all prime numbers in a range, using for example the Sieve of Eratosthenes (for large number this operation might take significant time, optionally you can display a **UIActivityIndicator** to notify the user computation is ongoing and/or use *DispatchQueue.global().async* to do the expensive operation on background)

4. one tab can use a Monte-Carlo simulation to compute the probabilistic distribution of the sum of a set of dices containing any number of dices having 4-faces, 6-faces, 8-faces, 10-faces, 12-faces and 20-faces (optionally: the

simulation can also take several seconds of time and similarly to the previous case you can show a **UIActivityIndicator** and use *DispatchQueue.global().async* to do the expensive operation on background). To generate a random number you can use *arc4random_uniform()*