Destiny

In this project you will make a 'choose your own adventure' style game, similar to the popular *Life Line* app on the App Store. You will get practice using a `struct` to manage the state of each "room."

1. Set up the UI.
    a. Create a label that will display the current 'room information' (a String, e.g. "You are in a dark and gloomy forest. You see two paths ahead of you…").
        i. The label should be centered on screen, its text should have center alignment, and long text should wrap to multiple lines.
            1. For the text to wrap, the label must have a set width (otherwise it will continue off the screen to the right). Luckily, you know how to do this programmatically.
        ii. Test that your label works before moving on.
    b. Create two buttons that will allow the user to choose one of two 'paths.'
        i. The buttons should be centered on screen, stacked vertically, and should take up the entire horizontal width minus a 10-pixel margin on each side.
    c. Create outlets for all UI elements.
    d. Create responder functions for the buttons; you'll complete these later.
2. Create the necessary data types.
    a. Think about the *states* of a CYOA style game – what does it need to remember? You need to know where you are, and where you can go.
    b. Create a `struct` called Story. Add the following properties to it:

```
let title:        String
let choice1:      String
let choice1index: Int
let choice2:      String
let choice2index: Int
```

    c. `choice1` and `choice2` are the 'paths' the user will choose, as text (and lead to the next Story). `choice1index` and `choice2index` will be explained later.
    d. Complete the `init` for Story.
    e. Next, create a `struct` Destiny that will serve as the 'game world'. Add the following code:

```
var currentStory : Int = 0

let stories = [
       Story(title: "this is the first story text (room) that will display",
       choice1: "first path choice", choice1index: 1, choice2: "second path
       choice", choice2index: 2),

       Story(title: "second room text", choice1: "third choice", choice1index:
       0, choice2: "fourth choice", choice2index: 1)
       Story(title: "final room", choice1: "the", choice1index: -1, choice2:
       "end", choice2index: -1)
]
```

Destiny stores the 'rooms' in the game as Story instances in a list called `stories`. The user's choice will lead to another Story (another index in the `stories` array). You don't need to fully understand Swift lists to complete this project (we will cover them in more depth later).

   f. Add functions to Destiny that will return the current story's `title`, and the current story's `choice1` and `choice2`.

3. Add property `var game = Destiny()` to the view controller.
   a. This is the 'brain' of the game, with a reference to the current story. It maintains the state of the game (with a Destiny object).
4. Update the label text to use the `title` of the current story.
5. Update the button text to the first room's choices.
6. Add code to handle user input.
   a. Each path chosen by the user should advance the story.
   b. Update the UI based on the current room.
7. Refactor your code to reduce redundancies. Use functions where something is done more than once.
8. Add more story elements (as Story objects) to the `stories` list to complete your game. Test thoroughly on a simulated device.

Extension Ideas

- Add a background image, suitable for your game. Use a UIImageView object that takes up the entire screen.
- Improve button visuals, e.g. colors that match the background image or rounded edges.
- Add *transitions*, such that when a new label / 'room' appears it doesn't just appear.