Spencer Goff
Assignment #4
July 2, 2017

1.

Description: Have the UITableView display two sections – one for items worth more than $50 and one for the rest.

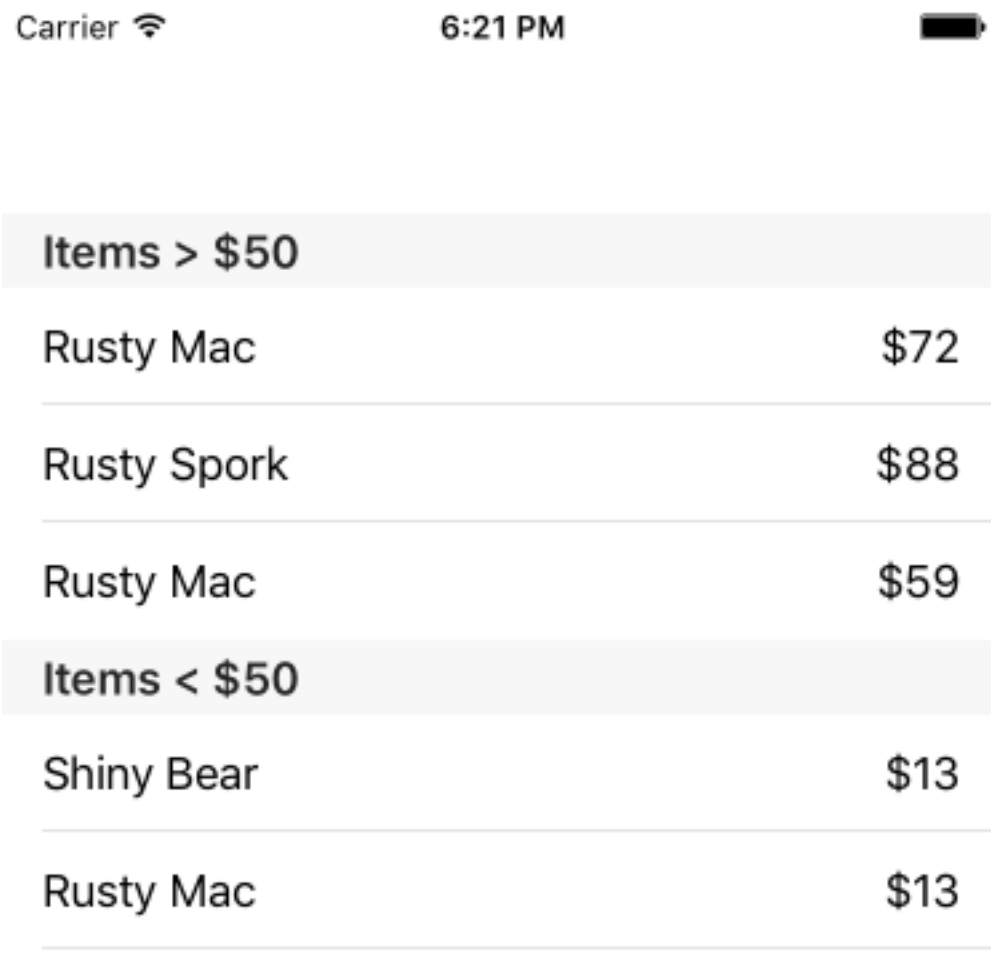
Logic: In ItemStore, I deleted "allItems" and instead made two separate arrays, one for items greater than or equal to $50 and one for items less than $50. Then, when an item is created, it is added to the appropriate array. In the view controller, I modified tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) to return thea appropriate row count based on which section the view is asking for. I also modified tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) in the same way, and added numberOfSections(in tableView: UITableView) to inform the view that the table will have 2 sections.

Input:

```
class ItemsViewController: UITableViewController {

    var itemStore: ItemStore!
    var sectionTitles = ["Items > $50", "Items < $50"]

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        if(section == 0) {
            return itemStore.itemsWorth50OrMore.count
        } else {
            return itemStore.itemsWorthLessThan50.count
        }
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell { //decides labels for each
        cell in UI
        let cell = tableView.dequeueReusableCell(withIdentifier: "UITableViewCell", for: indexPath) //get a new or recycled cell to save
            memory
        var item: Item
        if indexPath.section == 0 {
            item = itemStore.itemsWorth50OrMore[indexPath.row]
        } else {
            item = itemStore.itemsWorthLessThan50[indexPath.row]
        }

        cell.textLabel?.text = item.name
        cell.detailTextLabel?.text = "$\(item.valueInDollars)"

        return cell
    }

    override func numberOfSections(in tableView: UITableView) -> Int {
        return 2
    }

    override func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
        return sectionTitles[section]
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        /* makes sure the content doesn't overlap w. the status bar at top */
        let statusBarHeight = UIApplication.shared.statusBarFrame.height //height of status bar (at top of screen)
        let insets = UIEdgeInsets(top: statusBarHeight, left: 0, bottom: 0, right: 0)
        tableView.contentInset = insets
        tableView.scrollIndicatorInsets = insets
    }
}
```

```swift
class ItemStore { //note: this is a Swift base class since it doens't inherit from anything
    var itemsWorth50OrMore = [Item]()
    var itemsWorthLessThan50 = [Item]()

    @discardableResult func createItem() -> Item { //the result of this function can be ignored by the caller
        let newItem = Item(random: true) //creates a new item with random property values
        if newItem.valueInDollars >= 50 {
            itemsWorth50OrMore.append(newItem)
        } else {
            itemsWorthLessThan50.append(newItem)
        }
        //allItems.append(newItem)
        return newItem
    }
```

Output:



2.

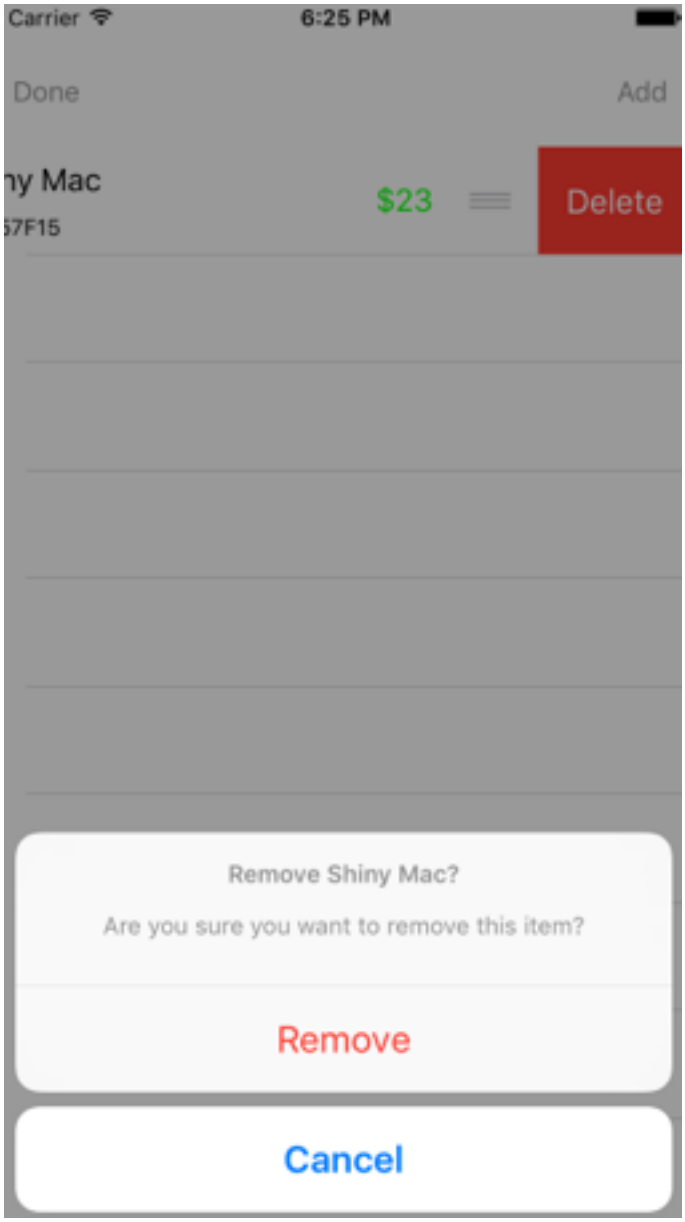
Description: When deleting a row, a confirmation button appears labeled Delete. Change the label of this button to Remove.

Logic: In the ItemsViewController, I modified tableView(_ tableView: UITableView, commit editingStyle: UITableViewCellEditingStyle, forRowAt indexPath: IndexPath), changing the "title" of "deleteAction" UIAlertAction from "Delete" to "Remove".

Input:

```
let deleteAction = UIAlertAction(title: "Remove", style: .destructive, handler: { (action)-
    > Void in
    self.itemStore.removeItem(item)
    self.tableView.deleteRows(at: [indexPath], with: .automatic)
})
```

Output:



3.

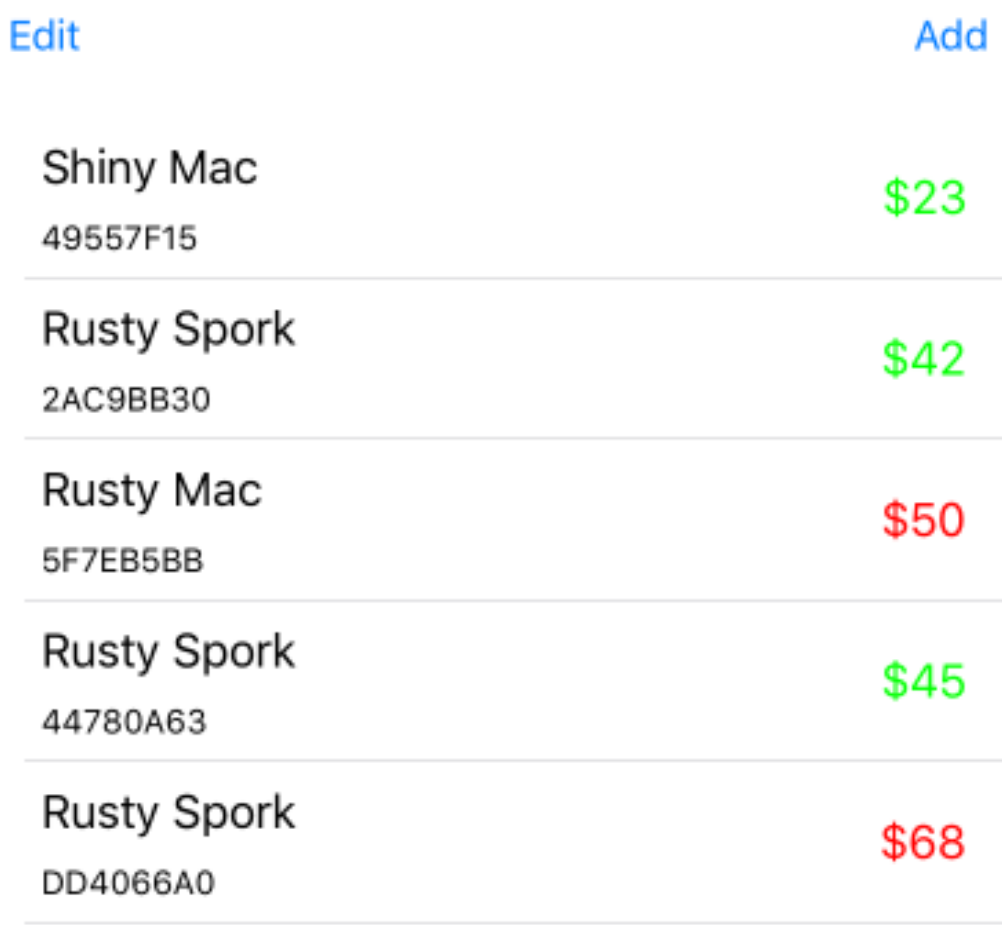
Description: Update the ItemCell to display the valueInDollars in green if the value is less than 50 and red if the value is greater than or equal to 50.

Logic: In ItemsViewController, in tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath), I added a statement to check if the item's value is >= 50. If it is, I set the value label's text color to red. Otherwise, I set it to green.

Input:

```
if item.valueInDollars >= 50 {
    cell.valueLabel.textColor = UIColor.red
} else {
    cell.valueLabel.textColor = UIColor.green
}
```

Output:

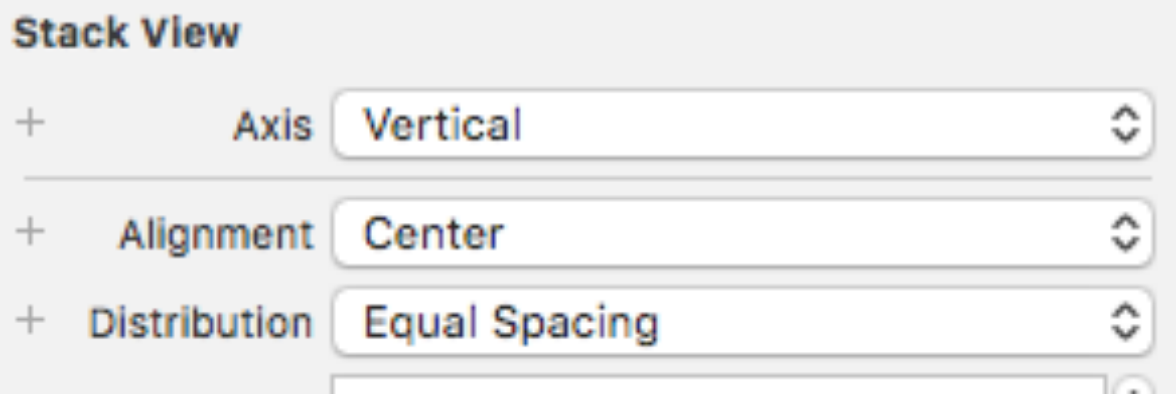


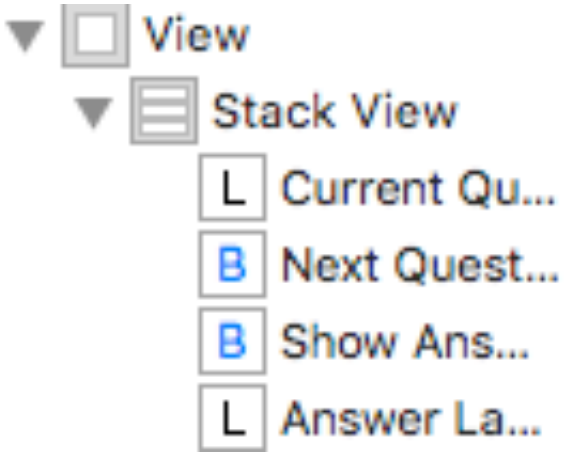
4.

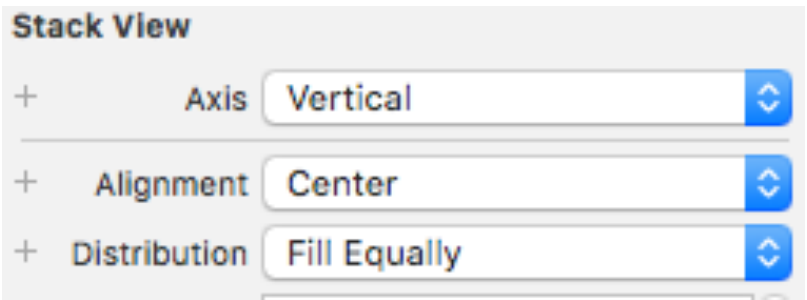
Description: Quiz and WorldTrotter are good candidates for using stack views. Update both of these applications to use UIStackView.

Logic: To use UIStackView in WorldTrotter, I added the new stack view to the "convert" view in Main.Storyboard. I added constraints so it takes up the entire screen, except for the status bar. I then made the existing labels subviews of this new stack view. This automatically removed their existing constraints. Next I opened the attributes inspector for the stack view, and changed the alignment to "center" and distribution to "equal spacing" so the labels and text field are spaced and centered nicely. I did the exact same thing for Quiz.

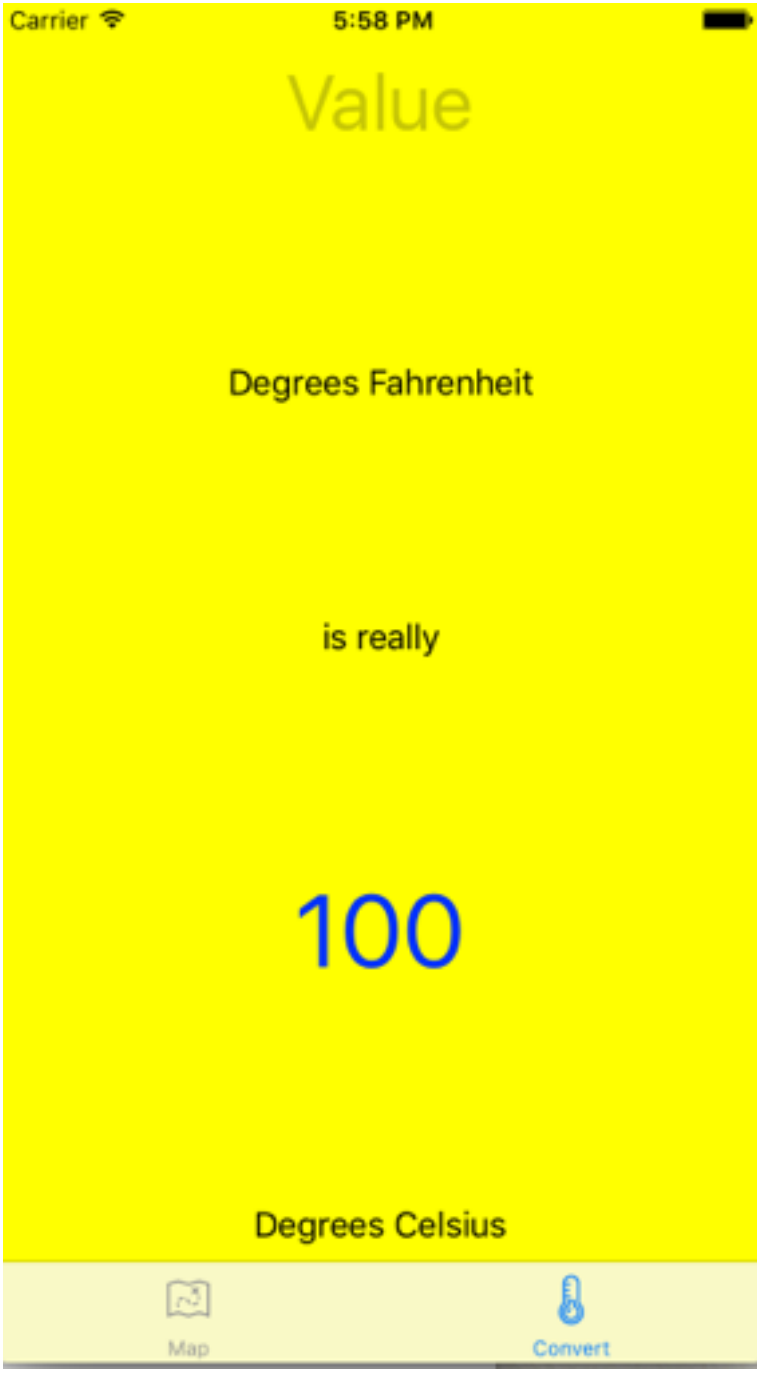
Input:

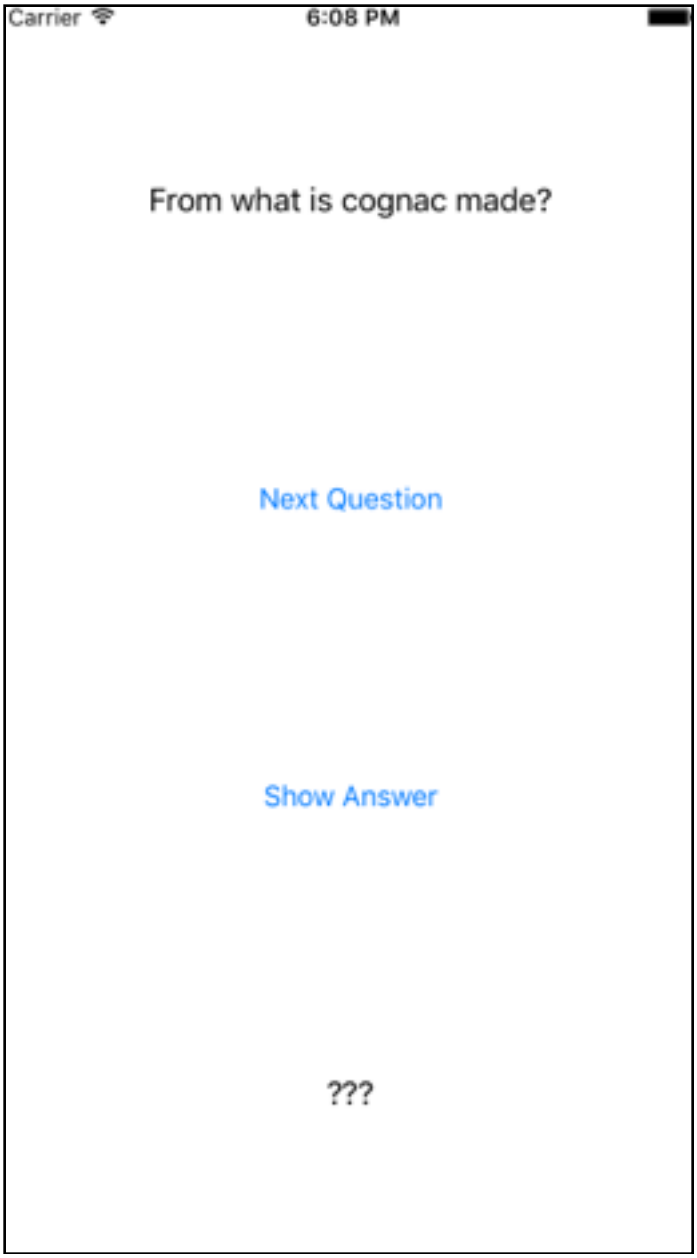






Output:


Carrier
5:58 PM
Value
Degrees Fahrenheit
is really
100
Degrees Celsius
Map
Convert


Carrier
6:08 PM
From what is cognac made?
Next Question
Show Answer
???