Create a New Project:

Step 1:

- Open Xcode
- If the previous project is opened, Click on `File->Close Project`
- Click or tap on "`File-> New -> Project…`"
- Click on `App`
- Product Name: **Tip Calculator**
- Organization Identifier: **Mobile Development**
- Interface: **Storyboard**
- Life Cycle: **UIKit App Delegate**
- Language: **Swift**
- Click the `Next` button

Step 2:

- Select the place to save it: I would recommend Desktop
- Click on `Create`

Step 3:

- In the project setting's `General` tab, scroll to the `Deployment Info` section, then for the `Device Orientation` ensure that only `Portrait` is selected. In landscape orientation, the numeric keypad would obscure parts of the Tip Calculator's UI.

Designing the Main Screen:

Step 1: Click on the `Main.storyboard` file

Step 2: Make sure that the device shown in the View matches the device specified in the Simulator (iPhone 11 Pro Max).

Step 3: Add a `Label` object

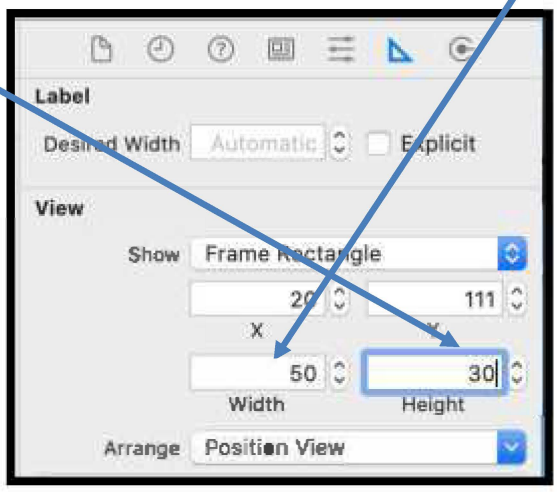
- Click the `Library`, select `Label` object and drag the object to the scene's upper-left corner, using the blue guidelines to position the Label at the recommended distance from the scene's top and left.
- Double click the `Label` and type **Enter Amount**, then press `Enter` to change its `Text` attribute.
- Re-arrange the `Label`, if needed, using the blue guide lines.

Step 4: Adding the `Label` that will display the formatted amount

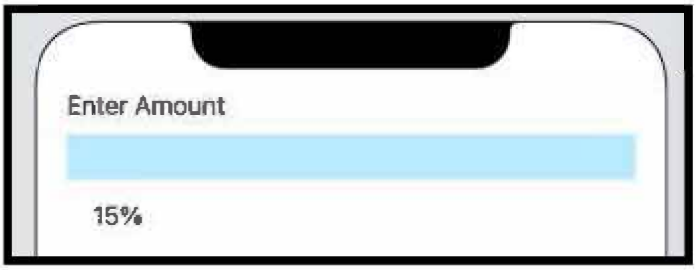
- Drag another `Label` below the Enter Amount label, such that the placement guides appear at the left margin and above the label
- Drag the middle sizing handle at the new `Label`'s right side until the blue guide line at the scene's right side appears (right margin)
- In the `Attributes` inspector, scroll to the `View` section and locate the `Label`'s `Background` attribute. Click the attribute's value, then select `Custom...` to display the `Colors` dialog. Using the `Crayon`'s tab, select the `Sky` (blue) crayon as the color, then set the `Opacity` to `50%` resulting in a lighter blue color.
- You can close the `Colors` dialog box.
- Using the bottom-center sizing handle of the `Label`, drag it to make the Height of the `Label` to be 30.
- With the Label selected, delete the value for its `Text` property in the Attributes inspector.
- Until this point, your app should look like this:

Step 5: Add another `Label` that displays the current custom Tip Percentage

- Drag another `Label` below the previous one using the blue guideline as a guide (left and upper).
- Double click and type **15%**, to change its `Text` property.
- Using the `Size` inspector (the 6th inspector), change the `Width` property to **50** and `Height` to **30**

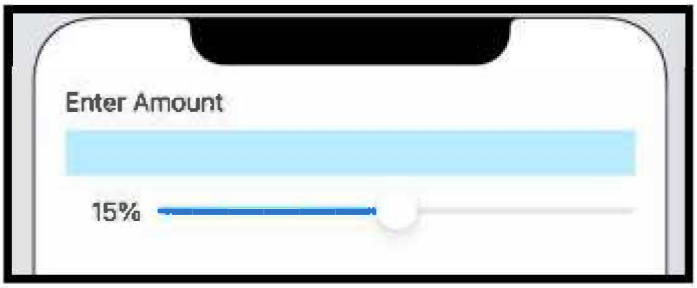


- Re-arrange the `Label`, if needed, using the blue guidelines.
- Using the `Attributes` inspector, change the alignment to right.
- Until this point, your app should look like this:



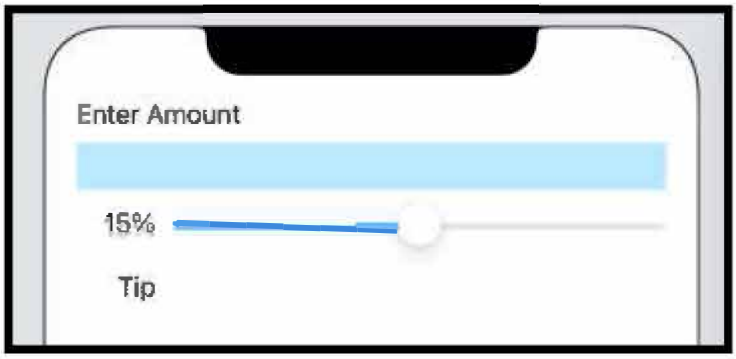
Step 6: Creating the Custom Tip Percentage Slider

- Drag a `Slider` from the Library onto the scene to the right of the previous label, using the blue guide lines as the recommended distance from the previous label and make sure that is centered vertically (upper and lower) with the previous label.
- Using the right sizing handle drag the control's right side until the blue guide line at the scene's right side appears (right margin).
- Use the `Attribute` inspector to set the Slider's `Minimum Value` to **0**, `Maximum Value` to **30**, and `Current Value` to **15**.
- Until this point, your app should look like this:



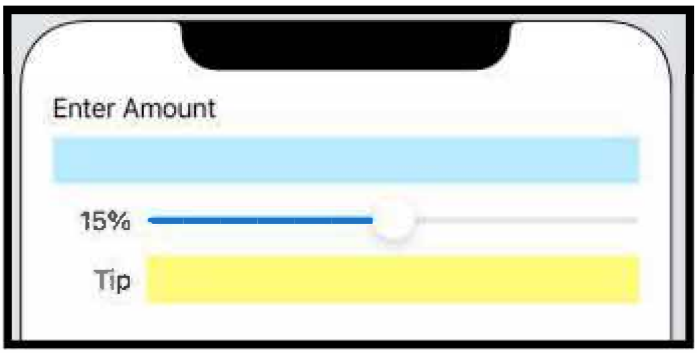
Step 7: Add another Label that displays the Tip

- Drag a `Label` object to the scene below the 15% label, using the left and upper blue line as guideline.
- Double-click the `Label` object and type **Tip**
- Using the `Size` inspector, change the `Width` property to **50** and `Height` to **30**
- Using the `Attributes` inspector, change the alignment to right.
- Until this point, your app should look like this:



Step 8: Add another `Label` that displays the formatted Tip amount

- Drag the `Label` to the right of the previous label following the blue guidelines (it should start at the same position as the Slider).
- Set the label to a `Height` of 30 and `Width` of about 316 (blue guide line at the scene's right side) using the sizing handle
- Use the `Attributes` inspector to clear the `Text` attribute, set the `Alignment` to center, and set the `Background` to `Banana` color.
- Set the `Autoshrink` property to `Minimum Font Scale` and change the value to **0.75** (If the text becomes too wide to fit in the Label, this will allow the text to shrink to 75% of its original font size to accommodate more text.
- Until this point, your app should look like this:



Step 9: Duplicate the last two labels to hold the Total of the bill

- Select both labels (Tip and Formatted Tip) by holding the *Shift* key and clicking each `Label`. Hold the *option* key and drag any one of the selected `Labels` down until the blue guides appears.
- Change the `Text` property of the left `Label` to **Total**
- Until this point, your app should look like this:

Step 10: Creating the `TextField` for receiving user input

- Drag a `TextField` from the Library to the bottom edge of the scene. (This text field will be hidden behind the numeric pad when the app first loads. You will receive the user's input through this `Text Field`, then format and display it in the blue `Label` at the top of the scene).
- Use the `Attributes` inspector to set its `Keyboard Type` to `Number Pad`, and `Keyboard Look` to `Dark`
- Your final app should look like this:

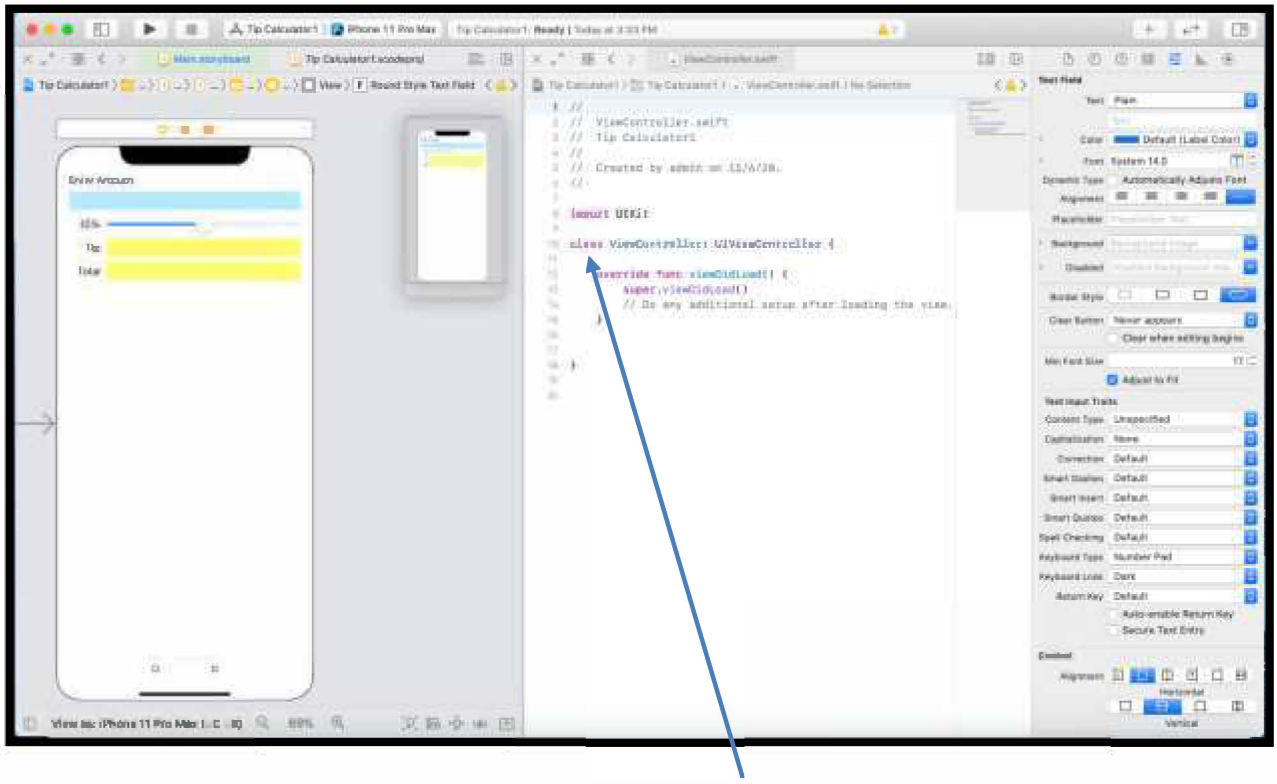
Creating Outlets with Interface Builder:

Step 1:

Using the Interface Builder, create the outlets for the UI components that the app interacts with programmatically.

To create the outlets, ensure that your scene's storyboard is displayed, and select the Assistant Editor

In order to have space for everything, please hide the left pane and the doc (component tree) area, so your screen should like the following:



Create the following outlets (Control and Drag) between the class definition and the first function:

- **billAmountLabel** (light blue label)
- **customTipPercentLabel** (15% - to the left of the Slider)
- **customTipPercentSlider** (slider)
- **tipCustomLabel** (first yellow label)
- **totalCustomLabel** (second yellow label)
- **inputTextField**

After creating those outlets, your code should be like this:

```swift
//
//  ViewController.swift
//  Tip Calculator1
//
//  Created by admin on 11/6/20.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var billAmountLabel: UILabel!
    @IBOutlet weak var customTipPercentLabel: UILabel!
    @IBOutlet weak var customTipPercentSlider: UISlider!
    @IBOutlet weak var tipCustomLabel: UILabel!
    @IBOutlet weak var totalCustomLabel: UILabel!
    @IBOutlet weak var inputTextField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }


}
```

Step 2: Checking the Outlets

- In the gray margin to the left of each outlet property is a small bullseye symbol indicating that the outlet is connected to a UI component.
- Hovering the mouse over that symbol highlights the connected UI component in the scene, so you can confirm that each outlet is connected properly.

Creating Actions with Interface Builder

You need to create actions (i.e. event handlers) that can respond to the user-interface events.

In the case of our app, we need to perform the following actions:

- When the user enters the bill amount (Text Field's Editing Changed event)
- When the user moves the Slider's thumb (Value Change event)

Step 1: Adding the `Text Field`'s action

- Control drag from the `Text Field` to the `Assistant` editor below the last function and before the last curly brace '}'
- Select `Action` from the popover (BE CAREFUL you need to change the event shown by default)
  - `Name`: **calculateTip**
  - `Event`: **select** `Editing Changed`
  - Click `Connect`

```
10  class ViewController: UIViewController {
11
12      @IBOutlet weak var billAmountLabel: UILabel!
13      @IBOutlet weak var customTipPercentLabel: UILabel!
14      @IBOutlet weak var customTipPercentSlider: UISlider!
15      @IBOutlet weak var tipCustomLabel: UILabel!
16      @IBOutlet weak var totalCustomLabel: UILabel!
17      @IBOutlet weak var inputTextField: UITextField!
18
19      override func viewDidLoad() {
20          super.viewDidLoad()
21          // Do any additional setup after loading the view.
22      }
23
24      @IBAction func calculateTip(_ sender: Any) {
25      }
26
27  }
28
29
```

Step 2: Adding the Slider's action

- The `calculateTip` action should also be called as the user changes the custom tip percentage.
- You can simply connect the `Slider` to this existing action to handle the `Slider`'s Value Changed event.
- Select the `Slider` in the scene, then hold the *control* key and drag from the `Slider` to the `calculateTip` method or func and release **(make sure that you release when the whole action is enclosed in box)**

- Now, if you hover the mouse over the bullseye of this action, you will see that is connected to two controls.

Che

Completing the ViewController.swift code:

Step 1: show the line numbers in the Assistant Editor:

- Click `XCode -> Preferences`
- Select `Text Editing` at the Top Bar
- Click on Show `Line Numbers`

Step 2: Override the method viewDidLoad:

- You typically override it to define tasks that can be performed only after the view has been initialized
- Add the following lines of code: lines 23t o 28:

```
19      override func viewDidLoad() {
20          super.viewDidLoad()
21          // Do any additional setup after loading the view.
22
23          //Reinforce that the initial tip percent is 15%
24          customTipPercentSlider.value = 15
25
26          //Select the TextField so keypad displays when the view loads
27          inputTextField.becomeFirstResponder()
28      }
```

Step 3: Code the calculateTip method

- Copy the following code; you are going to get errors on several statements that perform the formatting of the number to currency; we will define that function in Step 4.
- To understand a little bit better the code, please read the comments already placed in the code.
- You do not need to copy those comments in your app, but I will recommend you do it, so you can have a clear understanding of the code.
- The next figure shows the code with the errors that you should have until this moment.

```
@IBAction func calculateTip(_ sender: Any) {
    //Creation of variables to be used in this method or function
    var decimal100: Double
    decimal100 = 100.0
    var customPercent: Double
    var billAmount: Double
    var inputDouble: Double

    //get the user input and store it in a variable
    let inputString = inputTextField.text

    //convert the value entered (string) to a double
    inputDouble = Double((inputString! as NSString).floatValue)

    //convert slider value to a decimal number and store in a variable
    let sliderValue = Int(customTipPercentSlider.value)

    //convert the slider value set up by the user to percent
    customPercent = Double(sliderValue) / decimal100

    //Identify who did generate the event. Was the slider?
    if sender is UISlider
    {
        //the thumb has been moved by the user, so update the label with the new custom percent
        customTipPercentLabel.text = NumberFormatter.localizedString(from: NSNumber(value: customPercent), number:
            NumberFormatter.Style.percent)
    }

    //If there is a bill amount, calculte the new tip and total
    //Please put attention that there is a negation (!) in fornt of the control inputString!
    if !inputString!.isEmpty
    {
        //convert the billAmount to add the decimal point
        billAmount = inputDouble / decimal100
```

```
            //Now we check if the event was generated by the TextField
            if sender is UITextField
            {
                //Update the billAmount with the currency format
                billAmountLabel.text = " " + formatAsCurrency(number: billAmount)        Cannot find 'formatAsCurrency' in scope
            }

            //Now, we calculate and display the tip and the total with currency format
            let customTip = billAmount * customPercent
            tipCustomLabel.text = formatAsCurrency(number: customTip)                   Cannot find 'formatAsCurrency' in scope
            totalCustomLabel.text = formatAsCurrency(number: billAmount + customTip)    Cannot find 'formatAsCurrency' in scope
        }
        else
        {
            //We clear all the labels
            billAmountLabel.text = " "
            customTipPercentLabel.text = NumberFormatter.localizedString(from: NSNumber(value: customPercent), number:
                NumberFormatter.Style.percent)
            tipCustomLabel.text = " "
            totalCustomLabel.text = " "
        }
    }

}
```

Step 4: Add Global Utility Functions

- Copy the following code containing the format function used throughout the class at the end of the file (after the closing curly brace "}")
- After typing that function, you should have no errors at all. If you do, please contact me immediately so I can help you fix those errors before your submission.

```
//Global utility functions
//Convert a numberic value to a localized currency string
func formatAsCurrency(number: Double) -> String
{
    return NumberFormatter.localizedString(from: NSNumber(value: number), number: NumberFormatter.Style.currency)
}
```

Customizing the App

After you run the app, and you see that it is working perfectly, you might consider adding an image between the Total and soft keyboard. There is plenty of space for a very nice image… This is completely optional, and it will not impact the grade of this app.

Adding an Icon to the App

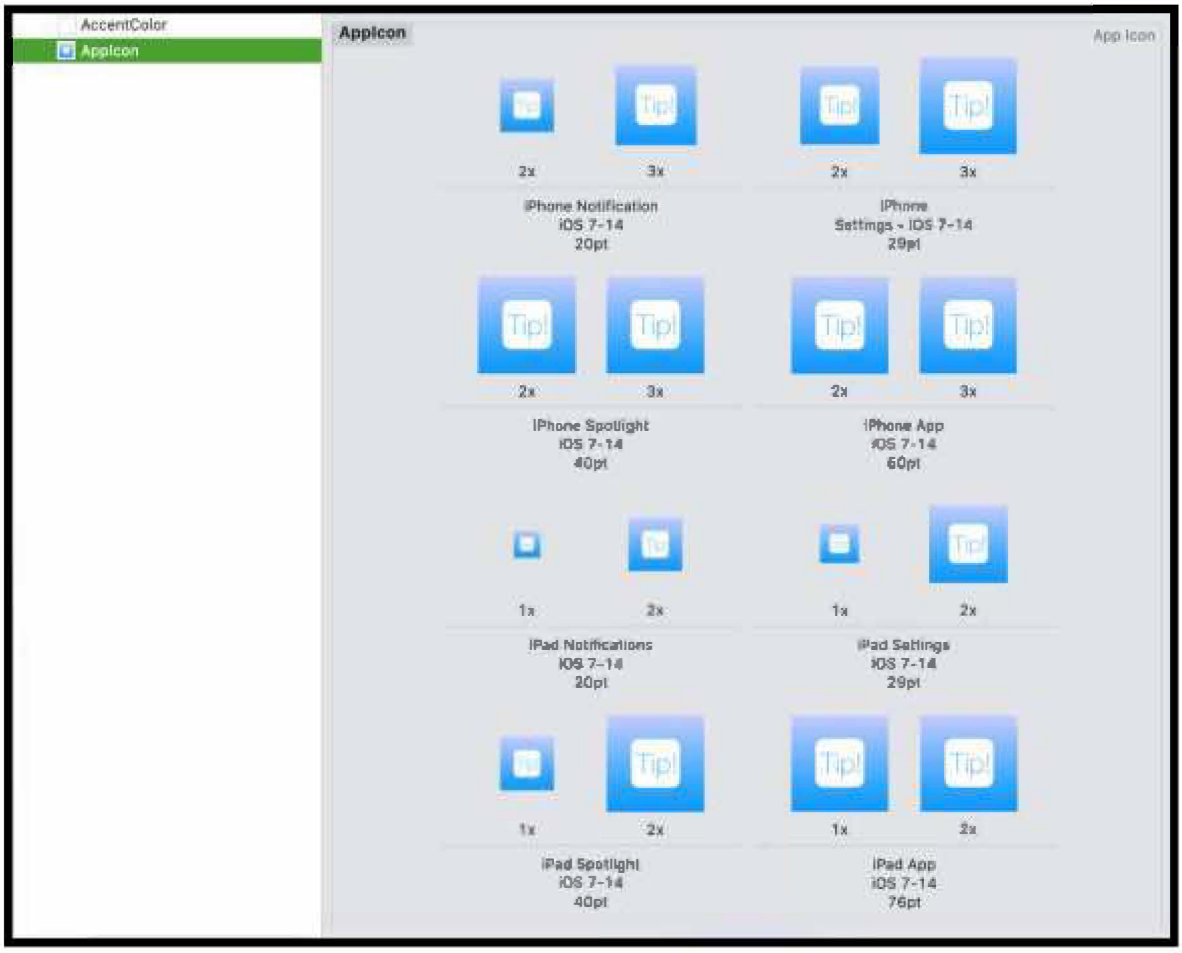
Please review Slides 29-31 from the first presentation, so you can remember how to add an App icon into the project.

I already create an icon, and all the Appicon resolutions and put them in a zip file located in the Lab 10 Assignment Drop Box. Follow these steps in order to add them to your project:

- Download the TipCalculator.zip file from Blackboard (Lab 10 Drop box)
- Using the Finder, click on the Downloads under Favorites in the left pane
- Double click the TipCalculatorApp.zip file. This will create an ios folder, and inside of it you will have three files and one folder (Appicon.appiconset).
- Click on the Appicon.appiconset folder
- Now, go to Xcode and click on the Assets.xcassets files under the left pane.
- Click on the AppIcon on the doc area.
- Now, you will start dragging from the Appicon.appiconset folder to the AppIcon placeholders in Xcode as follow:

Xcode AppIcon placeholder	Icon Image file
iPhone Notification 20 pt 2x	Icon-App-20x20@2x.png
iPhone Notification 20 pt 3x	Icon-App-20x20@3x.png
iPhone Settings 29 pt 2x	Icon-App-29x29@2x.png
iPhone Settings 29 pt 3x	Icon-App-29x29@3x.png
iPhone Spotlight 40 pt 2x	Icon-App-40x40@2x.png
iPhone Spotlight 40 pt 3x	Icon-App-40x40@3x.png
iPhone App 60 pt 2x	Icon-App-60x60@2x.png
iPhone App 60 pt 3x	Icon-App-60x60@3x.png
iPad Notifications 20 pt 1x	Icon-App-20x20@1x.png
iPad Notifications 20 pt 2x	Icon-App-20x20@2x.png
iPad Settings 29 pt 1x	Icon-App-29x29@1x.png
iPad Settings 29 pt 2x	Icon-App-29x29@2x.png
iPad Spotlight 40 pt 1x	Icon-App-40x40@1x.png
iPad Spotlight 40 pt 2x	Icon-App-40x40@2x.png
iPad App 76 pt 1x	Icon-App-76x76@1x.png
iPad App 76 pt 2x	Icon-App-76x76@2x.png
iPad Pro 83.5 pt 2x	Icon-App-83.5x83.5@2x.png
App Store 1024 pt 1x	iTunesArtwork@2x.png

- After you drag all the files, your AppIcon should look like this:



Submitting the App for grading

- Go to storage device and locate the folder that hast the name of the App (Tip Calculator).
- Right-click on it and select `Send to` and then `Compressed(zipped) folder`
- Edit the name of the zip file: **YourName_Lab10.zip** (FernandoPaniagua_Lab10.zip)
- Go to Blackboard and upload the zip file in the corresponding Assignment Drop Box.