```swift
//
//  UnitConverterViewController.swift
//  Kintematics Calculator
//
//  Created by Luke Deratzou on 3/3/18.
//  Copyright © 2018 Luke Deratzou. All rights reserved.
//

import UIKit

class UnitConverterViewController: UIViewController,
 UIPickerViewDelegate, UIPickerViewDataSource, UITextFieldDelegate{

    @IBOutlet weak var enterValueTextField: UITextField!
    @IBOutlet weak var convertBtn: UIButton!
    @IBOutlet weak var pastConversionBtn: UIButton!
    @IBOutlet weak var pastConversionView: UITextView!

    @IBOutlet weak var titleLabel: UILabel!
    @IBOutlet weak var returnBtn: UIButton!
    @IBOutlet weak var settingsBtn: UIButton!

    @IBOutlet weak var selectVarBtn: UIButton!

    @IBOutlet weak var selectUnitsToConvertToBtn: UIButton!
    @IBOutlet weak var correctSelectUnitBtn: UIButton! //good

    var varChoicePickerData : [String] = ["Select variable...",
     "Velocity", "Acceleration", "Distance", "Time", "Mass", "Force",
     "Energy"]
    var varChoiceString : String = "Select variable..."
    var convertFromUnit : String = "Select unit..."
    var convertToUnit: String = "Select unit..."

    var toPass: String = ""
    var listOfVars: [PhysicsVariable] = [PhysicsVariable]()

    //var pastConversionList: String = ""

    var bottomPickView: UIView!

    var exitHelpMode = false

    @IBOutlet weak var resultLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        formatButtonsAndLabels()
        if Helper.MODE == "Help" {
            pastConversionView.isHidden = true
            helpMode()
            return
        }
        let pastConversions =
         UserDefaults.standard.getListOfSavedConversions()
        if pastConversions.count == 0 {
            pastConversionBtn.isEnabled = false
        }
        enterValueTextField.delegate = self
        self.hideKeyboardWhenTappedAround()
        setUpConversion()
        // Do any additional setup after loading the view.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        if Helper.MODE == "Help" {
            if exitHelpMode {
                Helper.MODE = "Normal"
            }
            return
        }

        if segue.identifier == "settings" {
            let svc = segue.destination as! SettingsViewController
            svc.toPass = "unitconverter"
        } else {
            let svc = segue.destination as! OptionsViewController
            svc.toPass = toPass
        }
    }

    override func viewWillAppear(_ animated: Bool) {

    }

    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        self.view.endEditing(true)
        return false
    }

    func formatButtonsAndLabels() {
```

```
var cornerRadius: CGFloat = 10
if self.view.frame.width > 500 {
    cornerRadius = 25
}

titleLabel.layer.masksToBounds = true
titleLabel.layer.cornerRadius = cornerRadius
resultLabel.layer.masksToBounds = true
resultLabel.layer.cornerRadius = cornerRadius


if self.view.frame.width > 500 {

} else {
    enterValueTextField.font =
     enterValueTextField.font?.withSize(14)
}
let isIphoneX = Helper.IS_IPHONE_X()
let smallestDimension: CGFloat =
 CGFloat(UserDefaults.standard.getButtonSize())
if isIphoneX {
    returnBtn.frame = CGRect(x: returnBtn.frame.minX, y: 42,
     width: smallestDimension, height: smallestDimension)
    settingsBtn.frame = CGRect(x: settingsBtn.frame.minX, y:
     42, width: smallestDimension, height: smallestDimension)
} else {
    returnBtn.frame = CGRect(x: returnBtn.frame.minX, y:
     returnBtn.frame.minY, width: smallestDimension, height:
     smallestDimension)
    settingsBtn.frame = CGRect(x: settingsBtn.frame.minX, y:
     settingsBtn.frame.minY, width: smallestDimension, height:
     smallestDimension)
}

if convertBtn.frame.width / 120 > convertBtn.frame.height / 30 {
    let newWidth: CGFloat = convertBtn.frame.height * (120/30)
    convertBtn.frame = CGRect(x: self.view.frame.width/2 -
     newWidth/2, y: convertBtn.frame.minY, width: newWidth,
     height: convertBtn.frame.height)
} else {
    let newHeight: CGFloat = convertBtn.frame.width * (30/120)
    convertBtn.frame = CGRect(x: convertBtn.frame.minX, y:
     convertBtn.frame.minY, width: convertBtn.frame.width,
     height: newHeight)
}

if pastConversionBtn.frame.width / 200 >
 pastConversionBtn.frame.height / 35 {
```

```
            let newWidth: CGFloat = pastConversionBtn.frame.height *
             (200/35)
            pastConversionBtn.frame = CGRect(x: self.view.frame.width/2
             - newWidth/2, y: pastConversionBtn.frame.minY, width:
             newWidth, height: pastConversionBtn.frame.height)
        } else {
            let newHeight: CGFloat = pastConversionBtn.frame.width *
             (35/200)
            pastConversionBtn.frame = CGRect(x:
             pastConversionBtn.frame.minX, y:
             pastConversionBtn.frame.minY, width:
             pastConversionBtn.frame.width, height: newHeight)
        }

        setUpBottomView()
    }

    func setUpBottomView() {
        bottomPickView = UIView(frame: CGRect(x: 0, y:
         self.view.frame.height - Helper.GET_BOTTOM_VIEW_HEIGHT(),
         width: self.view.frame.width, height:
         Helper.GET_BOTTOM_VIEW_HEIGHT()))
        bottomPickView.backgroundColor = UIColor(displayP3Red: 93/255,
         green: 188/255, blue: 210/255, alpha: 1)
        bottomPickView.tag = 100
        self.view.addSubview(bottomPickView)
        bottomPickView.isHidden = true
    }
    //sets up a new conversion after view loads or when user selects
     for another conversion
    func setUpConversion() {
        resultLabel.isHidden = true
        pastConversionView.isHidden = true
        convertBtn.isEnabled = false
        correctSelectUnitBtn.isEnabled = false
        selectUnitsToConvertToBtn.isEnabled = false
        selectVarBtn.isEnabled = true
        enterValueTextField.text?.removeAll()
        varChoiceString = "Select variable..."
        convertFromUnit = "Select original unit..."
        convertToUnit = "Select new unit..."
        setUpBtnTitles()
        convertBtn.setBackgroundImage(UIImage(named:
         "button_convert.gif"), for: .normal)
        enterValueTextField.isEnabled = true
    }

    @IBAction func showSelectVarPicker(_ sender: UIButton) {
```

```swift
        showBottomView(tag: 0)
    }

    @IBAction func showSelectUnitPicker(_ sender: UIButton) {
        showBottomView(tag: (varChoicePickerData.firstIndex(of:
         varChoiceString) ?? 0))
    }

    @IBAction func showSelectToUnitPicker(_ sender: UIButton) {
        showBottomView(tag: (varChoicePickerData.firstIndex(of:
         varChoiceString) ?? 0) * -1)
    }

    func showBottomView(tag: Int) {
        bottomPickView.tag = 111
        for i in self.view.subviews {
            if i.tag != 111 {
                i.isUserInteractionEnabled = false
            }
        }
        self.bottomPickView.isHidden = false
        addPickerView(tag: tag)
        let doneBtn = DoneButton(frame: CGRect(x: self.view.frame.width
         - Helper.GET_DONE_BTN_WIDTH() - 5, y: 5, width:
         Helper.GET_DONE_BTN_WIDTH(), height:
         Helper.GET_DONE_BTN_HEIGHT()))
        //doneBtn.tag = sender.tag
        doneBtn.addTarget(self, action: #selector(hideBottomView), for:
         .touchUpInside)
        self.bottomPickView.addSubview(doneBtn)

    }

    func addPickerView(tag: Int) {
        let unitPicker: UIPickerView = UIPickerView(frame: CGRect(x: 0,
         y: 0, width: self.bottomPickView.frame.width, height:
         self.bottomPickView.frame.height))
        unitPicker.delegate = self
        unitPicker.dataSource = self
        unitPicker.tag = tag
        self.bottomPickView.addSubview(unitPicker)
    }

    @objc func hideBottomView(_ sender: AnyObject) {
        for i in self.view.subviews {
            if i.tag != 111 {
                i.isUserInteractionEnabled = true
            }
        }
        /*let c1 = !varChoiceString.contains("...") &&
         !varChoiceString.isEmpty
        if convertFromUnit.contains("...") && c1 {
            let newTitle = "Select \(varChoiceString) unit..."
            convertFromUnit = newTitle
            convertToUnit = newTitle
        }*/
        checkBtn()
        bottomPickView.isHidden = true
        for i in bottomPickView.subviews {
            if let viewWithTag = self.bottomPickView.viewWithTag(i.tag)
             {
                viewWithTag.removeFromSuperview()
            }
        }

        correctSelectUnitBtn.isEnabled = varChoiceString != "Select
         variable..." && varChoiceString != ""
        selectUnitsToConvertToBtn.isEnabled =
         correctSelectUnitBtn.isEnabled
        setUpBtnTitles()

    }

    func setUpBtnTitles() {
        selectVarBtn.setTitle(varChoiceString, for: .normal)
        selectUnitsToConvertToBtn.setTitle(convertToUnit, for: .normal)
        correctSelectUnitBtn.setTitle(convertFromUnit, for: .normal)
    }
    //depricated
    func createLabels() {
        //tag -100: first label (select var)
        //tag -200: second label (select first unit) (converting from)
        //tag -300: third label (select second unit) (convert to)
        for i in view.subviews {
            if i.tag < -99 {
                if let viewWithTag = self.view.viewWithTag(i.tag) {
                    viewWithTag.removeFromSuperview()
                }
            }
        }
        var fontSize:CGFloat = 0
        switch true {
        case self.view.frame.height > 600 && self.view.frame.width <
         500:
            fontSize = 15
        case self.view.frame.width > 500:
            fontSize = 40
        default:
            fontSize = 13
        }
        let labelHeight: CGFloat = 50
        let labelWidth: CGFloat = 120
        if varChoiceString != "" && varChoiceString != "Select
         variable..." {
            let label: UILabel = UILabel(frame: CGRect(x:
             selectVarBtn.frame.maxX + 10, y: selectVarBtn.frame.minY,
             width: labelWidth, height: labelHeight))
            label.text = varChoiceString
            label.font = UIFont(name: "Menlo", size: fontSize)
            label.textColor = UIColor.white
            label.tag = -100
            self.view.addSubview(label)
        } else {
            return
        }
        if convertFromUnit != "" && convertFromUnit != "Select unit..."
         {
            let label: UILabel = UILabel(frame: CGRect(x:
             correctSelectUnitBtn.frame.maxX + 10, y:
             correctSelectUnitBtn.frame.minY, width: labelWidth,
             height: labelHeight))
            label.text = convertFromUnit
            label.font = UIFont(name: "Menlo", size: fontSize)
            label.textColor = UIColor.white
            label.tag = -200
            self.view.addSubview(label)
        } else {
            return
        }
        if convertToUnit != "" && convertToUnit != "Select unit..." {
            let label: UILabel = UILabel(frame: CGRect(x:
             selectUnitsToConvertToBtn.frame.maxX + 10, y:
             selectUnitsToConvertToBtn.frame.minY, width: labelWidth,
             height: labelHeight))
            label.text = convertToUnit
            label.font = UIFont(name: "Menlo", size: fontSize)
            label.textColor = UIColor.white
            label.tag = -300
            self.view.addSubview(label)
        }

    }

    @IBAction func valueToConvertEditingEnd(_ sender: UITextField) {
```

```swift
    checkBtn()
}

@IBAction func returnToEquationButton(_ sender: UIButton) {
    //delete
}
//somewhere in this function, find way to right away save the
 conversion into the defaults thingy.
@IBAction func convertButtonAction(_ sender: UIButton) {
    if convertBtn.backgroundImage(for: .normal) == UIImage(named:
     "button_another-conversion.gif") {
        setUpConversion()
    } else {
        pastConversionBtn.isEnabled = true
        if varChoiceString == "Select variable..." {
            let errorAlert = UIAlertController(title: "Error!",
             message: "Select a variable for conversion.",
             preferredStyle: .alert)
            let errorAlertAction = UIAlertAction(title: "Got it!",
             style: .cancel, handler: { (ACTION: UIAlertAction) in
            })
            errorAlert.addAction(errorAlertAction)
            present(errorAlert, animated: true)
            return
        } else if !isItAValidNumber(input:
         enterValueTextField.text!) {
            let errorAlert = UIAlertController(title: "Error!",
             message: "Input a valid value.", preferredStyle:
             .alert)
            let errorAlertAction = UIAlertAction(title: "Got it!",
             style: .cancel, handler: { (ACTION: UIAlertAction) in
            })
            errorAlert.addAction(errorAlertAction)
            present(errorAlert, animated: true)
            convertBtn.isEnabled = false
            return
        }
        resultLabel.isHidden = false
        let convertedValue = Helper.CONVERT_UNITS(from:
         convertFromUnit, to: convertToUnit, value:
         Double(enterValueTextField.text!)!)
        let roundedConvertedValue =
         RoundByDecimals.ROUND_BY_DECIMALS(value:
         "\(convertedValue)")
        //bad stuff below
        /*let tempVar: PhysicsVariable = PhysicsVariable.init(name:
         PhysicsVariable.FIX_NAME(varName: varChoiceString))
        tempVar.value = Double(enterValueTextField.text!)!
        tempVar.unConvertedValue = tempVar.value
        tempVar.unit = convertFromUnit
        let temp = Helper.CONVERT_UNITS(physicsVar: tempVar, toSI:
         true)
        tempVar.value = temp
        listOfVars.append(tempVar)
        resultLabel.text = "\(tempVar.getRealName()): \(temp)
         \(tempVar.getSIUnits())"*/

        resultLabel.text = "\(varChoiceString):
         \(roundedConvertedValue) \(convertToUnit)"

        convertBtn.setBackgroundImage(UIImage(named:
         "button_another-conversion.gif"), for: .normal)
        selectVarBtn.isEnabled = false
        correctSelectUnitBtn.isEnabled = false
        selectUnitsToConvertToBtn.isEnabled = false
        enterValueTextField.isEnabled = false
        //Fixed code:
        let newConversion: String = "\(varChoiceString):
         \(roundedConvertedValue) \(convertToUnit) (converted from
         \(enterValueTextField.text!) \(convertFromUnit))"
        setUpPastConversions(newConversion: newConversion)

        //FIX THIS CODE! E: delete!

        //pastConversionList += "\n" + "\n" +
         "<\(tempVar.getRealName()): \(tempVar.value)
         \(tempVar.getSIUnits()) (converted from \(tempVar.unit))"
    }
}

func isItAValidNumber(input: String) -> Bool {
    return Double(input) != nil
}

@IBAction func showPastConversionAction(_ sender: UIButton) {
    if pastConversionBtn.backgroundImage(for: .normal) ==
     UIImage(named: "button_hide-past-conversions.gif") {
        pastConversionHideOrSeek(showPastConv: false)
        pastConversionBtn.setBackgroundImage(UIImage(named:
         "button_show-past-conversions.gif"), for: .normal)
        pastConversionBtn.frame = CGRect(x:
         pastConversionBtn.frame.minX, y:
         pastConversionBtn.frame.minY -
         pastConversionBtn.frame.height*1.25, width:
         pastConversionBtn.frame.width, height:
         pastConversionBtn.frame.height)
    } else {
        let pastConversions =
         UserDefaults.standard.getListOfSavedConversions()
        //insert code for loading old conversions here...
        pastConversionBtn.frame = CGRect(x:
         pastConversionBtn.frame.minX, y:
         pastConversionBtn.frame.minY +
         pastConversionBtn.frame.height*1.25, width:
         pastConversionBtn.frame.width, height:
         pastConversionBtn.frame.height)
        var formattedForViewConversions: String = ""
        var endedAThing = false
        for i in 0...pastConversions.count-1 {
            if i == 0 || endedAThing {
                endedAThing = false
                formattedForViewConversions.append("•")
                if i == 0 {
                    formattedForViewConversions.append(" ")
                }
            }
            let index =
             pastConversions.index(pastConversions.startIndex,
             offsetBy: i)
            if pastConversions[index] == "," {
                formattedForViewConversions += "\n"
                endedAThing = true
            } else {

                 formattedForViewConversions.append(pastConversions
                 [index])
            }
        }

        pastConversionView.text = formattedForViewConversions
        pastConversionView.font = UIFont(name: "Menlo", size:
         Helper.GET_FONT_SIZE())
        pastConversionHideOrSeek(showPastConv: true)
        pastConversionBtn.setBackgroundImage(UIImage(named:
         "button_hide-past-conversions.gif"), for: .normal)
    }

}

/*func numberOfSavedConversions() -> Int {
    let listOfSavedConversions =
     UserDefaults.standard.getListOfSavedConversions()
    var commaCount = 0
    for i in 0...listOfSavedConversions.count-1 {
        let index =
         listOfSavedConversions.index(listOfSavedConversions
         .startIndex, offsetBy: i)
        if listOfSavedConversions[index] == "," {
            commaCount += 1
        }
    }
    return commaCount
}*/

func setUpPastConversions(newConversion: String) {
    var stuffToSave: String =
     UserDefaults.standard.getListOfSavedConversions()
    if !stuffToSave.isEmpty {
        stuffToSave += ", "
    }
    stuffToSave += newConversion
    UserDefaults.standard.setListOfSavedConversions(value:
     stuffToSave)
    Helper.CONFIGURE_SAVED_CONVERSIONS()

}

func pastConversionHideOrSeek(showPastConv: Bool) {
    pastConversionView.isHidden = !showPastConv
    resultLabel.isHidden = showPastConv
    enterValueTextField.isHidden = showPastConv
    convertBtn.isHidden = showPastConv
    titleLabel.isHidden = showPastConv
    selectVarBtn.isHidden = showPastConv
    correctSelectUnitBtn.isHidden = showPastConv
    selectUnitsToConvertToBtn.isHidden = showPastConv
    for i in view.subviews {
        if i.tag < -99 {
            if let viewWithTag = self.view.viewWithTag(i.tag) {
                viewWithTag.removeFromSuperview()
            }
        }
    }
}

// The number of columns of data
func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return 1
}

// The number of rows of data
```

```
func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent
 component: Int) -> Int {
    if pickerView.tag == 0 {
        return varChoicePickerData.count
    } else {
        let tag:Int = Int(pickerView.tag.magnitude) - 1
        return Helper.LIST_OF_UNIT_LISTS[tag].count
    }
}
// **Need to find out how to change # of rows based on type of
 data...**
// The data to return for the row and component (column) that's
 being passed in
func pickerView(_ pickerView: UIPickerView, titleForRow row: Int,
 forComponent component: Int) -> String? {
    if pickerView.tag == 0 {
        return varChoicePickerData[row]
    } else {
        let tag:Int = Int(pickerView.tag.magnitude) - 1
        return Helper.LIST_OF_UNIT_LISTS[tag][row]
    }
}

// Capture the picker view selection
func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int,
 inComponent component: Int) {

    if pickerView.tag == 0 {
        varChoiceString = varChoicePickerData[row]
    } else if pickerView.tag.signum() == -1 {
        let tag:Int = Int(pickerView.tag.magnitude) - 1
        convertToUnit = Helper.LIST_OF_UNIT_LISTS[tag][row]
    } else {

        convertFromUnit =
         Helper.LIST_OF_UNIT_LISTS[pickerView.tag-1][row]
    }

    // This method is triggered whenever the user makes a change to
     the picker selection.
    // The parameter named row and component represents what was
     selected.
}

func pickerView(_ pickerView: UIPickerView, viewForRow row: Int,
 forComponent component: Int, reusing view: UIView?) -> UIView {
    var pickerLabel: UILabel? = (view as? UILabel)
    var fontSize: CGFloat = 15
```

```
        switch true {
        case self.view.frame.height > 600 && self.view.frame.width <
         500:
            fontSize = 15
        case self.view.frame.width > 500:
            fontSize = 40
        default:
            fontSize = 13
        }
        if pickerLabel == nil {
            pickerLabel = UILabel()
            pickerLabel?.font = UIFont(name: "Menlo", size: fontSize)
            pickerLabel?.textAlignment = .center
        }
        pickerLabel?.textColor = UIColor.white
        if pickerView.tag == 0 {
            pickerLabel?.text = varChoicePickerData[row]
        } else {
            let tag:Int = Int(pickerView.tag.magnitude) - 1
            pickerLabel?.text = Helper.LIST_OF_UNIT_LISTS[tag][row]
        }
        return pickerLabel!
    }

    func pickerView(_ pickerView: UIPickerView, rowHeightForComponent
     component: Int) -> CGFloat {
        switch true {
        case self.view.frame.height > 600 && self.view.frame.width <
         500:
            return 22.0
        case self.view.frame.width > 500:
            return 48.0
        default:
            return 22.0
        }

    }

    func checkBtn() {
        let con1 = !convertFromUnit.isEmpty &&
         !convertFromUnit.contains("...")
        let con2 = !(enterValueTextField.text?.isEmpty)!
        let con3 = !convertToUnit.isEmpty &&
         !convertToUnit.contains("...")
        if con1 && con2 && con3 {
            convertBtn.isEnabled = true
        } else {
            convertBtn.isEnabled = false
```

```swift
    }
}

//text field restrictions:
func textField(_ textField: UITextField, shouldChangeCharactersIn
 range: NSRange, replacementString string: String) -> Bool {
    let inverseSet =
     NSCharacterSet(charactersIn:"0123456789").inverted
    let components = string.components(separatedBy: inverseSet)
    let filtered = components.joined(separator: "")

    if filtered == string {
        return true
    } else {
        if string == "." {
            let countdots =
             textField.text!.components(separatedBy:".").count - 1
            if countdots == 0 {
                return true
            } else {
                if countdots > 0 && string == "." {
                    return false
                } else {
                    return true
                }
            }
        } else {
            if string == "-" {
                let countNegs =
                 textField.text!.components(separatedBy:"-").count
                 - 1
                if countNegs <= 1 {
                    return true
                } else {
                    if countNegs > 1 && string == "-" {
                        return false
                    } else {
                        return true
                    }
                }
            } else {
                if string == "e" {
                    let countdots =
                     textField.text!.components(separatedBy:"e")
                     .count - 1
                    if countdots == 0 {
                        return true
                    } else {
                        if countdots > 0 && string == "e" {
                            return false
                        } else {
                            return true
                        }
                    }
                } else {
                    return false
                }
            }
        }
    }
}

func helpMode() {
    //could use something to tell users that they are in Help
     Mode...
    //like a label at the top or whereever it would fit that says
     help mode
    //or it can be a banner at top that can just be dismissed with
     an x... idk
    disableEverything()
    addHelpModeBtns()
    setUpInvisibleBtns()
}

func addHelpModeBtns() {

    var factor: CGFloat = 1
    if self.view.frame.width > 500 {
        factor = 2
    }

    let helpView = UIView(frame: CGRect(x: 0, y:
     self.view.frame.maxY - 50*factor, width:
     self.view.frame.width*factor, height: 50*factor))
    helpView.backgroundColor = UIColor.gray
    self.view.addSubview(helpView)
    let leftArrow: UIButton = UIButton(frame: CGRect(x: 50*factor,
     y: self.view.frame.maxY - 50*factor, width: 50*factor, height:
     50*factor))
    leftArrow.setBackgroundImage(UIImage.init(named:
     "left_arrow.png"), for: .normal)
    leftArrow.addTarget(self, action: #selector(prevView), for:
     .touchUpInside)
    self.view.addSubview(leftArrow)
```

```
        let rightArrow: UIButton = UIButton(frame: CGRect(x:
         self.view.frame.maxX - 100*factor, y: self.view.frame.maxY -
         50*factor, width: 50*factor, height: 50*factor))
        rightArrow.setBackgroundImage(UIImage.init(named:
         "right_arrow.png"), for: .normal)
        rightArrow.addTarget(self, action: #selector(nextView), for:
         .touchUpInside)
        self.view.addSubview(rightArrow)

        let exitBtn: UIButton = UIButton(frame: CGRect(x:
         self.view.frame.midX - factor*75/2, y: self.view.frame.maxY -
         40*factor, width: 75*factor, height: 25*factor))
        exitBtn.setBackgroundImage(UIImage(named:
         "button_exit-help.gif"), for: .normal)
        //later add a nice picture for this (or just copy the one from
         quiz)
        exitBtn.addTarget(self, action: #selector(exitHelp), for:
         .touchUpInside)
        self.view.addSubview(exitBtn)
    }

    func disableEverything() {
        for i in self.view.subviews {
            i.isUserInteractionEnabled = false
        }

    }

    func setUpInvisibleBtns() {
        var listOfBtns: [UIButton] = [UIButton]()
        listOfBtns.append(UIButton(frame: titleLabel.frame))
        listOfBtns.append(UIButton(frame: returnBtn.frame))
        listOfBtns.append(UIButton(frame: settingsBtn.frame))

        listOfBtns.append(UIButton(frame: CGRect(x: 0, y: 0, width:
         selectVarBtn.frame.width, height: selectVarBtn.frame.maxY)))
        listOfBtns.append(UIButton(frame: CGRect(x:
         correctSelectUnitBtn.frame.minX, y: 0, width:
         correctSelectUnitBtn.frame.width, height:
         correctSelectUnitBtn.frame.maxY)))
        listOfBtns.append(UIButton(frame: enterValueTextField.frame))
        listOfBtns.append(UIButton(frame:
         selectUnitsToConvertToBtn.frame))
        listOfBtns.append(UIButton(frame: convertBtn.frame))
        listOfBtns.append(UIButton(frame: resultLabel.frame))
        listOfBtns.append(UIButton(frame: pastConversionBtn.frame))

        for i in 0...listOfBtns.count-1 {
            listOfBtns[i].tag = i
            listOfBtns[i].backgroundColor = UIColor.clear
            listOfBtns[i].addTarget(self, action: #selector(openPopup),
             for: .touchUpInside)
            self.view.addSubview(listOfBtns[i])
        }
    }

    @objc func openPopup(_ sender: UIButton) {
        if popUpAlreadyExists() {
            closePopup(self)
            return
        }

        var factor:CGFloat = 1
        if self.view.frame.width > 500 {
            factor = 2.5
        }
        let popUp: UITextView = UITextView(frame: CGRect(x:
         self.view.frame.midX-120*factor, y: self.view.frame.midY -
         90*factor, width: 240*factor, height: 180*factor))
        popUp.text = HelpPopups.UNITCONVERT[sender.tag]
        popUp.tag = -64
        popUp.isEditable = false
        popUp.backgroundColor = UIColor(displayP3Red: 93/255, green:
         188/255, blue: 210/255, alpha: 1)
        popUp.font = UIFont(name: "Menlo", size: Helper.GET_FONT_SIZE()
         + 1*factor)
        self.view.addSubview(popUp)
        let exitGesture = UITapGestureRecognizer(target: self, action:
         #selector(closePopup))

        self.view.addGestureRecognizer(exitGesture)
        /*tag:
         0: titleLabel (select calculator)
         1: returnbtn
         2: showowkrview
         3: previous showwork
         4: next showwork
         5: page number
         */
    }

    func popUpAlreadyExists() -> Bool {
        for i in self.view.subviews {
            if i.tag == -64 {
                return true
            }
```

```swift
        }
        return false
    }

    @objc func closePopup(_ sender: Any) {
        for i in self.view.subviews {
            if i.tag == -64 {
                if let viewWithTag = self.view.viewWithTag(i.tag) {
                    viewWithTag.removeFromSuperview()
                }
            }
        }
    }

    @objc func exitHelp(_ sender: UIButton) {
        exitHelpMode = true
        performSegue(withIdentifier: "settings", sender: self)
    }

    @objc func nextView(_ sender: UIButton) {
        performSegue(withIdentifier: "practice problems", sender: self)
        //move to next view
    }
    @objc func prevView(_ sender: UIButton) {
        performSegue(withIdentifier: "show equation", sender: self)
    }

}
```