COMP 1601A: Mobile Application Development

Carleton University
Winter 2021 Midterm Exam Solutions

1. [8] Answer the following questions about lines 4–8 from **Conversions.swift**:

```
4       var convFrom = "From Type"
5       var convTo = "To Type"
6       var convert: (_ from: Double) -> Double?
7
8       func formatConversion(_ from: Double) -> String {
```

   (a) [1] In one word, what type is `convFrom`?
   A: String

   (b) [1] In one word, what type is `formatConversion`?
   A: function or closure

   (c) [1] In one word, what type is `convert`?
   A: function or closure

   (d) [1] What does _ do on line 8?
   A: Allows arguments to formatConversion to not need an argument label for the from: function argument.

   (e) [1] What does _ do on line 6?
   A: Allows arguments to convert to not need an argument label for the from: function argument.

   (f) [1] What is the `->` do on line 6?
   A: It indicates that the convert function has a return value (of type Double?).

   (g) [2] What does the `?` on line 6 indicate? Briefly explain what it means.
   A: The ? indicates that the returned type (a Double) is an optional type, meaning it can have a value of Double or nil.

2. [2] When is the `else` clause in `formatConversion()` (**Conversions.swift**, lines 8–14) executed? Be precise.
   A: It is executed when the call to self.convert() returns nil, i.e., when the conversion has failed.

3. [2] The values of the `conversions` dictionary (**Conversions.swift**, lines 28–40) are generated by a call to a `Converter()` function. What code is being run by this call? How do you know?
   A: The init() method (function) of Converter is being called. We know this because init() is the only constructor for Converter, and a constructor is always called when creating a struct, if such a constructor has been defined.

4. [2] Assume we have a program that includes **Conversions.swift**. We're given `ks` of type String which we are assured contains a value in kilometers. How can we convert `ks` to `m`, a `Double` which contains the number of miles equal to the number of kilometers in `ks`? Your code should be as concise as possible given the functionality of the included code. Be sure to declare `m` as a constant.

   A: let m = conversions["km to mi"]!.convert(Double(ks)!)!

5. [6] Answer the following about `theImage` in picviewer-1's **Contentview.swift**:

   (a) [1] Where is `theImage` declared?
   A: theImage is declared on line 11 in the ContentView struct and again on line 43 in the ActiveImage struct. The first is a @State declaration, meaning this is where storage is allocated to store the value; the second is a @Binding declaration, which means it is a reference to the @State declaration. (Full credit for just mentioning both lines of declaration or for just mentioning the @State declaration.)

   (b) [1] What is the value of `theImage` used for, precisely?
   A: It is used to store the name of the image asset to be displayed. It is used by the Image view on line 50 in ActiveImage to display the image.

   (c) [2] Where in the code is the value of `theImage` changed? When is this code called?
   A: theImage's value is changed on lines 26 and 30, in the ContentView View struct. This code is called when an item is selected from the Animals! menu at the top of the screen (it is in the callback function for each defined menu item button).

   (d) [2] As written, can the value of `theImage` be read inside of `tapReset()` (lines 63-67)? Why or why not?
   A:Yes, t can be read because it has been locally declared as a binding inside the ActiveImage view. You can access it by prepending it with self (i.e., self.theImage) but actually this isn't strictly needed.

6. [5] Answer the following about lines 76–84 of picviewer-1's **Contentview.swift**:

```
76      var magnifying: some Gesture {
77          MagnificationGesture().onChanged { amount in
78              self.currentAmount = amount - 1
79          }
80          .onEnded { amount in
81              self.finalAmount += self.currentAmount
82              self.currentAmount = 0
83          }
84      }
```

   (a) [1] When is line 78 called?
   A: This function is called repeatedly as the user moves their finger as part of a magnification gesture.

   (b) [1] How would the behavior of the program change if we replaced `self.currentAmount` with `self.finalAmount` on line 78?
   A: If we used finalAmount, then the magnification gesture would work as before, except that subsequent magnification gestures would start at the initial zoom level rather than allowing for continued magnification.

   (c) [1] This code declares `magnifying`. Where is it used?
   A: magnifying is used on line 57 to register one of two simultaneous gestures (magnifying and rotating, both of which are done with two fingers).

   (d) [2] Which is run more frequently, line 78 or lines 81 and 82? Why?
   A: Line 78 is run much more frequently, as 78 is run as the gesture is in progress, repeatedly, while 81 and 82 are only run once the gesture ends, i.e., when the user lifts up their fingers.

7. [7] Answer the following about lines 53–54 of picviewer-1's **Contentview.swift**:

```
.position(moved ? position :
          CGPoint(x: g.size.width / 2, y: g.size.height / 2))
```

   (a) [1] What are the possible values of `moved`?
   A: true or false (it is a Bool)

   (b) [2] What would happen if we just replaced this code with `.position(position)`? Why?
   A: It would work fine, except the image would start in the top left because position is initialized to 0,0.

   (c) [2] Explain at a high level the logic of the argument of `.position` here. How does it determine the position of the image?
   A: If moved is false, it determines the position from the g object returned from geometry reader (which basically specifies a rectangle the View is being placed in), setting the position to the middle of g vertically and horizontally. If true, then it just uses the position state.

   (d) [2] There are two declarations of `moved`. Do they both have to use the exact same name? Explain.
   A: They do not have to use the same name. We can have one name in ContentView and another in ActiveView. If we change the name in ActiveView to be different, we have to change the @Binding declaration on line 44, change line 34 to use the new name in front of tho colon (but keep the old name with the dollar sign, as this refers to the moved inside of ContentView), and change the references to moved on lines 53, 65, 89, and 93. (For full credit, must say no and make sure to distinguish between the two moved's on line 34.)

End of exam. Code listings begin on the next page.

Converter2/Shared/Conversions.swift

```
//  Conversions.swift

struct Converter {
    var convFrom = "From Type"
    var convTo = "To Type"
    var convert: (_ from: Double) -> Double?

    func formatConversion(_ from: Double) -> String {
        if let to = self.convert(from) {
            return "\(from) \(self.convFrom) is \(to) \(self.convTo)."
        } else {
            return "Converting \(from) \(convFrom) to \(self.convTo) failed."
        }
    }

    init(_ from: String, _ to: String, _ f: @escaping (_ from: Double) -> Double?) {
        self.convFrom = from
        self.convTo = to
        self.convert = f
    }
}

func InToCM(_ inch: Double) -> Double {
    let cm = 2.54 * inch
    return cm
}

let conversions: [String: Converter] =
    ["F to C": Converter("Farenheit", "Celsius",
                         {F in return ((F - 32.0)*(5/9))}),
     "C to F": Converter("Celsius", "Farenheit",
                         {C in return ((9/5)*C + 32.0)}),
     "km to mi": Converter("kilometers", "miles",
                           {k in
                            let m = k * 0.6213712
                            return m
                           }),
     "inch to cm": Converter("inches", "centimeters",
                             InToCM)
    ]
```

picviewer-1/picviewer-1/ContentView.swift

```
//
//  ContentView.swift
//  picviewer-1
//
//  Created by Anil Somayaji on 2/15/21.
//

import SwiftUI

struct ContentView: View {
    @State private var theImage = "kittens"
    @State private var moved = false
    @State private var finalAmount: CGFloat = 1
```

```swift
    @State private var angle = Angle(degrees: 0.0)

    func resetState() {
        moved = false
        finalAmount = 1
        angle = Angle(degrees: 0.0)
    }

    var body: some View {
        VStack {
            Menu("Animals!") {
                Button("Kittens", action: {
                    theImage = "kittens"
                    resetState()
                })
                Button("Sad Dog", action: {
                    theImage = "sadDog"
                    resetState()
                })
            }
            ActiveImage(theImage: $theImage, moved: $moved, finalAmount:
                $finalAmount, angle: $angle)
        }
    }
}

struct ActiveImage: View {
    @State private var position = CGPoint(x: 0, y: 0)
    @State private var currentAmount: CGFloat = 0

    @Binding var theImage: String
    @Binding var moved: Bool
    @Binding var finalAmount: CGFloat
    @Binding var angle: Angle

    var body: some View {
        GeometryReader {g in
            Image(theImage)
                .resizable()
                .scaledToFit()
                .position(moved ? position :
                            CGPoint(x: g.size.width / 2, y: g.size.height / 2))
                .scaleEffect(finalAmount + currentAmount)
                .gesture(dragging)
                .gesture(SimultaneousGesture(magnifying, rotating))
                .rotationEffect(self.angle)
                .onTapGesture(count: 1, perform: tapReset)
        }
    }

    func tapReset() {
        self.finalAmount = 1
        self.moved = false
        self.angle = Angle(degrees: 0.0)
    }

    var rotating: some Gesture {
        RotationGesture()
            .onChanged { angle in
                self.angle = angle
            }
    }

    var magnifying: some Gesture {
        MagnificationGesture().onChanged { amount in
            self.currentAmount = amount - 1
        }
        .onEnded { amount in
            self.finalAmount += self.currentAmount
            self.currentAmount = 0
        }
    }

    var dragging: some Gesture {
        DragGesture()
            .onChanged {s in
                self.moved = true
                self.position = s.location
            }
            .onEnded {s in
                self.moved = true
                self.position = s.location
            }
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```