COMP 1601A: Mobile Application Development

Carleton University
Winter 2021 Midterm Exam

March 3, 2021

There are seven questions on **2 pages** (+3 code listing pages) worth a total of 32 points. Answer all questions in a text file following the supplied template. If you find a question is ambiguous, explain your interpretation and answer the question accordingly.

This test is open book. You may use previous tutorials and assignments, your notes, and any generally available online resources. However, you **may not collaborate, work with, or get assistance from anyone**, including but not limited to anyone in this class. *Remember there will be selected post-exam interviews where you may be asked how you came up with your answers.*

Be sure to monitor the Announcements channel on Teams, as clarifications or corrections to the exam will be posted there. Do not use any other channels on Teams (or any other communication methods) during the exam to discuss anything related to the exam, except to talk with the instructor or TAs.

Cite any sources that you use that were not part of or referenced directly referenced in class. For example, you do not need to cite the official Swift documentation, but you do need to cite a random stack overflow answer if you used it to come up with your answer. *No matter the source, the words you use should be your own.*

You have 80 minutes. Good luck!

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`?

   (b) [1] In one word, what type is `formatConversion`?

   (c) [1] In one word, what type is `convert`?

   (d) [1] What does _ do on line 8?

   (e) [1] What does _ do on line 6?

   (f) [1] What is the -> do on line 6?

   (g) [2] What does the `?` on line 6 indicate? Briefly explain what it means.

2. [2] When is the `else` clause in `formatConversion()` (**Conversions.swift**, lines 8–14) executed? Be precise.

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?

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.

5. [6] Answer the following about `theImage` in picviewer-1's **Contentview.swift**:
   (a) [1] Where is `theImage` declared?
   (b) [1] What is the value of `theImage` used for, precisely?
   (c) [2] Where in the code is the value of `theImage` changed? When is this code called?
   (d) [2] As written, can the value of `theImage` be read inside of `tapReset()` (lines 63-67)? Why or why not?

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?
   (b) [1] How would the behavior of the program change if we replaced `self.currentAmount` with `self.finalAmount` on line 78?
   (c) [1] This code declares `magnifying`. Where is it used?
   (d) [2] Which is run more frequently, line 78 or lines 81 and 82? Why?

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`?
   (b) [2] What would happen if we just replaced this code with `.position(position)`? Why?
   (c) [2] Explain at a high level the logic of the argument of `.position` here. How does it determine the position of the image?
   (d) [2] There are two declarations of `moved`. Do they both have to use the exact same name? Explain.

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()
    }
}
```