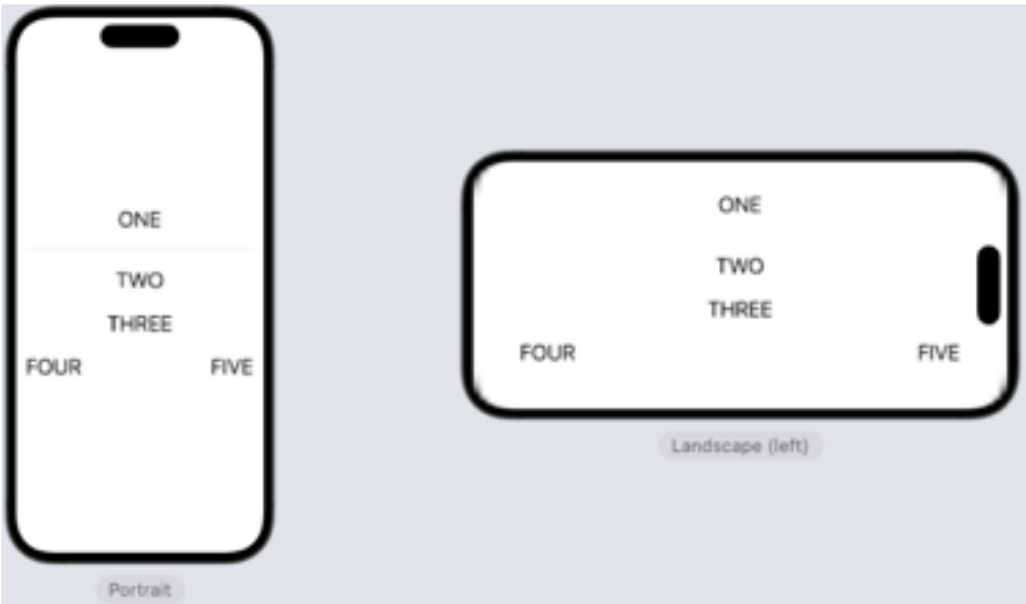
M.C.A. (3rd Semester)

CS8028: Mobile Application Development using iOS

Question Bank

Unit: 1 Basic SwiftUI Views and Controls

1. Compare SwiftUI with UIKit framework. Explain any four advantages of SwiftUI.
2. Write Swift code and design sample user interface using ZStack. Draw the preview user interface and explain the solution.
3. Design SwiftUI user interface using Label and Image view. Use at least two different modifiers for Label and Image and explain your code.
4. Create user interface using Toggle and Stepper. Write code snippet, draw diagram, and explain your solution.
5. Write SwiftUI code to design following user interface using VStack, HStack, Text, Divider, and Spacer.



6. How Swift prevents common programming errors? Explain any five.
7. Create user interface using Image view with resizable, scaleToFill, frame, clipShape, and opacity modifiers. Write code snippet, draw diagram and explain your solution.
8. Write code snippet and explain how to apply group of styles using ViewModifier.

9. Draw the user interface designed by “Code Block-1” and “Code Block-2”. Compare and discuss the output.

```swift
var body: some View {
    // Code Block-1
    VStack (spacing: 28) {
        Text("ONE")
        Divider()
        Text("TWO")
        Text("THREE")
        HStack {
            Text("FOUR")
            Spacer()
            Text("FIVE")
        }
    }
    .font(.largeTitle)
    .padding()
}
```

```swift
var body: some View {
    // Code Block-2
    VStack (spacing: 28) {
        Text("ONE")
        Text("TWO")
        Text("THREE")
        HStack {
            Text("FOUR")
            Spacer()
            Divider()
            Text("FIVE")
        }
    }
    .font(.largeTitle)
    .padding()
}
```

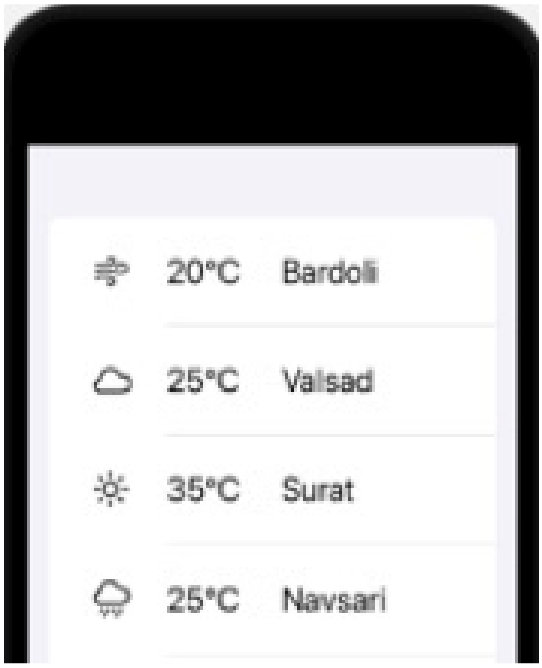
10. What is optional type in Swift? Justify its usage with example.
11. Explain custom parameter attribute that constructs views from closures using appropriate example and code snippet.
12. Which view is used to display one or more lines of read-only text? Design sample user interface and explain that view with lineLimit and truncationMode modifiers.
13. Write two different syntaxes for declaring an array in Swift. With appropriate example and code snippet, explain array of integer.
14. Write Swift code snippet and explain Text Field and Text Editor with appropriate example.
15. With appropriate example, explain how to add SwiftUI to legacy UIKit app.

Unit: 2 List Views, Scroll Views, and Advanced Components

1. What is the use of List view? Compare List and ScrollView. Discuss with appropriate examples.
2. Which functionalities are available in edit mode of a List? Explain editable list with appropriate example and code snippet.
3. Analyze the following code snippet and explain it.

```
List {
    ForEach(searchResults, id: \.self) { food in
        LabeledContent(food.name) { Text("\(food.category.rawValue)")
        }
    }
    .searchable(text: $searchText,
                prompt: "Search Test")
    .searchScopes($scope) {
        Text("All").tag(FruitSearchScope.all)
        Text("Fruit").tag(FruitSearchScope.food(.fruit))
        Text("Meat").tag(FruitSearchScope.food(.meat))
        Text("Vegetable").tag(FruitSearchScope.food(.vegetable))
    }
    .navigationTitle("Foods")
```

4. Explain use of expanding list and disclosure group with appropriate example and necessary code snippet.
5. Write SwiftUI code to display weather information using List view as given below:



6. Explain editable List with code snippet and appropriate example.
7. What is lazy stack? Compare LazyHStack, LazyVStack with HStack, VStack. Explain one advantage and one limitation of lazy stack.
8. What is SwiftUI widget? How it is useful? Explain three key components of SwiftUI widget.
9. Explain how to use custom rows in a List view with necessary code snippet and example.
10. Create editable collection of TodoTask which allows user to update TodoTask. Write Swift code for required type and view. Explain your solution.
11. Which SwiftUI lazy view is used to display tabular content? Give example and explain with necessary

code snippet.

12. Step-by-step explain, how to create SwiftUI widget for existing app.

Unit: 3 Navigation

1. What is NavigationStack? Considering appropriate example, write Swift code to implement static navigation. Explain your solution.
2. Explain the use of NavigationPath with appropriate example and code snippet.
3. Explain NavigationStack and NavigationLink with appropriate example and code snippet.
4. Design solution using init(sidebar:detail:) and demonstrate multi-column navigation. Write necessary code snippet and explain.
5. What is TabView? Write code snippet and explain how to use it.
6. Design solution using init(sidebar:content:detail:) and demonstrate multicolumn navigation. Write necessary code snippet and explain.
7. Write necessary Swift code and justify the following statement:
   "Apple provides the NavigationPath struct, as a type-erased list of data representing the content of a navigation stack."
8. How to navigate among multiple unrelated views? Explain with appropriate example and code snippet.

Unit: 4 SwiftUI with Data

1. What is property wrapper? Explain any one binding property wrapper with appropriate example and code snippet.
2. Explain @Binding property wrapper using following code snippet.

```
struct ClassRoomLight: View {
    @State private var lightsOn = false
    var body: some View {
        VStack(spacing: 28) {
            Button("Change Light from Parent") {
                lightsOn.toggle()
            }
            Text(lightsOn ? "Parent Lights ON!" :
                "Parent Lights OFF..")
            Toggle("Child Lights", isOn: $lightsOn)
                .padding()
        }
        .font(.largeTitle)
```

3. When to use @ObservedObject property wrapper? Write code snippet and explain.
4. How to use LocationManager in iOS app? Write code snippet and explain major steps.
5. Analyze the following code snippet and explain it.

```
// Create the state.
@State private var isPlaying: Bool = false
var body: some View {
    VStack (spacing: 28) {
        Toggle("isPlaying", isOn: $isPlaying)
        Divider()
        HStack {
            // Read the state.
            Button(isPlaying ? "Pause" : "Play") {
                // Write the state.
                isPlaying.toggle()
            }
            Image(systemName: isPlaying ?
                "pause.circle.fill" :
                "play.circle.fill")
```

6. Compare @State and @Binding. Write code snippet and discuss when one is more suitable then other.
7. "Use @ObservedObject when you have a custom type you want to use that might have multiple properties and methods, or might be shared across multiple views." Explain this with appropriate code snippet.
8. Write necessary code snippet and explain the use of @EnvironmentObject.
9. Write necessary code snippet and explain the use of @Binding.
10. "@Binding property wrapper creates two-way binding." Explain this with code snippet.
11. Analyze the following code snippet and explain it.

```swift
class MyCounter: ObservableObject {
    @Published var count = 0
}
struct CounterView: View {
    @ObservedObject var counter : MyCounter
    var body: some View {
        Button("Increase Count from CounterView") {
            counter.count += 1
        }
    }
}
struct ContentView: View {
    @StateObject var counter = MyCounter()
    var body: some View {
        VStack(spacing: 28) {
            Button("Decrease Count from ContentView") {
                counter.count -= 1
            }
            Text("Current count is: \(counter.count)")
            CounterView(counter: counter)
```

12. When to use @Observable macro? Write code snippet and explain.

Unit: 5 Combine and Firebase

1. What is Combine? Explain concept of publisher and subscriber.
2. Which design pattern is popularly used with reactive framework? Draw the diagram and compare MVC with MVVM.
3. What is form validation? How to validate a form using Combine? Write code snippet and discuss.
4. "When a client subscribes to a publisher, the subscription should be stored in a set of AnyCancellable." Explain this with code snippet.
5. Analyze the following code snippet and explain it.

```swift
class StopWatchTimer: ObservableObject {
    @Published var deciseconds: Int = 0
    @Published var seconds: Int = 0
    @Published var minutes: Int = 0
    @Published var started = false

    private var cancellableSet: Set<AnyCancellable> = []

    func start() {
        deciseconds = 0
        seconds = 0
        minutes = 0

        cancellableSet.store {
            Timer.publish(every: 0.1, on: RunLoop.main, in:
                .default)
                .autoconnect()
                .sink { [self] _ in
                    deciseconds = (deciseconds + 1) % 10
            }
```

6. Write appropriate code snippet and explain debounce(for:scheduler:options:) and map(_:).
7. Analyze the following code snippet and explain it.

```swift
class SignupViewModel: ObservableObject {
    // Input
    @Published var username = ""
    @Published var password = ""
    @Published var confirmPassword = ""

    // Input validation
    @Published var isValid = false
    @Published var usernameMessage = " "
    @Published var passwordMessage = " "

    private var cancellableSet: Set<AnyCancellable> = []

    init() {
        usernameValidPublisher
            .receive(on: RunLoop.main)
            .map { $0 ? " " : "Username must be at least 6
                characters long" }
            .assign(to: \.usernameMessage, on: self)
            .store(in: &cancellableSet)
```

8. Write appropriate code snippet and explain dataTaskPublisher(for:) and sink(receiveValue:).

9. Analyze the following code snippet and explain it.

```
let forecastURL = URL(string:
    "https://api.openweathermap.org/data/2
    .5/forecast?lat=\(latitude)&lon=\(longitude)&appid=\(apiKey)&units=metric")!
URLSession
    .shared
    .dataTaskPublisher(for: URLRequest(url: forecastURL))
    .map(\.data)
    .decode(type: ForecastWeather.self, decoder: decoder)
    .receive(on: RunLoop.main)
    .sink { completion in
        switch completion {
        case .finished:
            break
        case .failure(let error):
            self.errorMessage = error.localizedDescription
        }
    } receiveValue: {
        self.forecast = $0.list
    }
    .store(in: &cancellableSet)
```

10. Step by step explain, Sign In with Apple in SwiftUI.
11. Step-by-step explain, how to use Firebase “RemoteConfig” in SwiftUI app.
12. Which Firebase and Google packages are required for using “Google Sign In” in iOS app? Step-by-step explain how to add those SPM packages in Xcode project and configure Firebase project.

Unit: 6 Core Data and Swift Data

1. What is Core Data? Explain its usage in iOS app development.
2. With appropriate example and code snippet, explain filtering Core Data objects with predicate.
3. With appropriate example and code snippet, explain deleting Core Data object.
4. Compare Core Data and SwiftData. Explain any two advantages of SwiftData.
5. Analyze the following code snippet and explain it.

```
var body: some View {
    NavigationStack {
        List(contacts, id: \.self) {
            ContactView(contact: $0)
        }
        .listStyle(.plain)
        .navigationTitle("Contacts")
        .navigationBarTitleDisplayMode(.inline)
        .toolbar {
            Button {
                isAddContactPresented.toggle()
            } label: {
                Image(systemName: "plus")
                    .font(.headline)
            }
        }
        .sheet(isPresented: $isAddContactPresented) {
            AddNewContact()
                .environmentObject(coreDataStack)
```

6. Analyze the following code snippet and explain it.

```
import Foundation
import CoreData

class CoreDataStack {
    private let persistentContainer: NSPersistentContainer
    var managedObjectContext: NSManagedObjectContext {
        persistentContainer.viewContext
    }

    init(modelName: String) {
        persistentContainer = {
            let container = NSPersistentContainer(name: modelName)
            container.loadPersistentStores { description, error in
                if let error = error {
                    print(error)
                }
            }
            return container
        }()
```