**Development Scenario 2: Medication Reminder App**

**Day 1: iOS Introduction and Setup**

**Task 1:** Set up Xcode and review the Swift language basics, focusing on syntax and control flow.

1. **Install Xcode:**
   * Download and install the latest version of Xcode from the Mac App Store.
2. **Set Up Xcode:**
   * Open Xcode and go through the initial setup steps.
   * Ensure that you have the latest version of the iOS SDK installed.
3. **Review Swift Basics:**
   * Open a new playground in Xcode:
     + Select "File" > "New" > "Playground."
     + Choose a blank template and click "Next."
     + Save the playground file.
   * Write basic Swift code to understand the following concepts:
     + Variables and constants
     + Data types and type inference
     + Control flow (if-else, switch, loops)
     + Functions and closures

**Task 2: Start a new project for a medication reminder app, establishing the basic navigation and view structure**

**1. Function with Default Parameters**

func increment(\_ value: Int, by: Int = 1) -> Int {

return value + by

}

print(increment(2))

// Output: 3

**Explanation:**

* **Purpose:** This function increments an integer by a specified amount. If no amount is specified, it increments by 1.
* **Parameters:**
  + **value**: The base value to increment.
  + **by**: The amount to increment by, with a default value of 1.
* **Function Call:**
  + **increment(2)** passes **2** to **value**, and since **by** has a default value of **1**, it increments **2** by **1** to return **3**.
* **Special Syntax:**
  + The **\_** before **value** allows the function to be called without a label for the first parameter.

**2. Variadic Parameters**

func makeSum(\_ num: Int...) -> Int {

var sum = 0

for x in num {

sum += x

}

return sum

}

**print("Sum: ", makeSum(10, 20, 40, 5)) // Output: Sum: 75**

**Explanation:**

* **Purpose:** This function calculates the sum of an arbitrary number of integers.
* **Parameters:**
  + **num**: A variadic parameter, meaning it can take zero or more integers.
* **Function Body:**
  + Initializes a **sum** variable to **0**.
  + Iterates over each integer in **num**, adding each to **sum**.
  + Returns the final **sum**.
* **Function Call:**
  + **makeSum(10, 20, 40, 5)** sums **10**, **20**, **40**, and **5** to return **75**.

**3. Finding Largest and Smallest Elements in an Array**

func sg(arr: [Int]) -> (largest: Int, smallest: Int) {

var largest = arr[0]

var smallest = arr[0]

for i in arr[1..<arr.count] {

if i < smallest {

smallest = i

} else if i > largest {

largest = i

}

}

return (largest, smallest)

}

let result = sg(arr: [3, 4, 5, 6])

print("largest: \(result.largest), smallest: \(result.smallest)") // Output: largest: 6, smallest: 3

**Explanation:**

* **Purpose:** This function finds the largest and smallest numbers in an array of integers.
* **Parameters:**
  + **arr**: An array of integers.
* **Function Body:**
  + Initializes **largest** and **smallest** to the first element of **arr**.
  + Iterates through the rest of the array:
    - Updates **smallest** if the current element is smaller.
    - Updates **largest** if the current element is larger.
  + Returns a tuple containing **largest** and **smallest**.
* **Function Call:**
  + **sg(arr: [3, 4, 5, 6])** returns the largest (**6**) and smallest (**3**) numbers in the array.

**4. Recursive Function**

swift

Copy code

func factorial(num: Int) -> Int {

if num == 0 || num == 1 {

return 1

} else {

return num \* factorial(num: num - 1)

} }

print(factorial(num: 5))

// Output: 120

**Explanation:**

* **Purpose:** This function calculates the factorial of a number using recursion.
* **Parameters:**
  + **num**: The integer for which the factorial is to be calculated.
* **Function Body:**
  + Checks if **num** is **0** or **1**. If true, returns **1** (base case).
  + Otherwise, returns **num** multiplied by the factorial of **num - 1** (recursive case).
* **Function Call:**
  + **factorial(num: 5)** computes **5 \* 4 \* 3 \* 2 \* 1** to return **120**.