IdeasLab Interview Answers

1. How do I investigate leaks and performance issues?

To investigate leaks and performance issues XCode provides several tools we can use. I usually use Leaks on the instrument, Memory Graph, and a simple deinit debugger on suspected objects.

On XCode instrument, there are several tools we can use to our apps performance. For memory leaks, we use Leaks tools. While the instruments monitor the app, we navigate throughout the app and see if the instrument detects leaks during that time. If it does, we can see what object is causing the leaks. After that, we can check the debug memory graph to make sure the leaks are located on the parts we suspect there are. The leaks can be seen on the sidebar. Since leaks are usually caused by strong reference causing an object not deinitialized properly. Sometimes the simplest method is the best method. I put a print function on deinit to see if the suspected object is really deinitialized or not.

Memory Leaks Example

In this example, we try to simulate memory leaks. We have two objects with strong references on one another.

```swift
class MainClass {

    var subs: [SubClass] = []

    func add(sub: SubClass) {
        subs.append(sub)
    }

    deinit {
        print("\(Self.self) object was deallocated")
    }
}

class SubClass {
    var main: MainClass?

    init (main: MainClass) {
        self.main = main
        self.main?.add(sub: self)
    }

    deinit {
        print("\(Self.self) object was deallocated")
    }
}
```

For the main viewcontroller, it's a simple VC with a button to navigate to the second view controller.

```
class ViewController: UIViewController {

    lazy var button: UIButton = {
        let button = UIButton(type: .system)
        button.setTitle("Navigate", for: .normal)
        button.addTarget(self, action: #selector(buttonTapped), for:
.touchUpInside)
        button.translatesAutoresizingMaskIntoConstraints = false
        return button
    }()

    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = .white
        view.addSubview(button)
        button.centerXAnchor.constraint(equalTo:
view.centerXAnchor).isActive = true
        button.centerYAnchor.constraint(equalTo:
view.centerYAnchor).isActive = true
    }

    @objc func buttonTapped() {
        if let navigate = self.navigationController {
            navigate.pushViewController(SecondViewController(), animated:
true)
        }
    }

}
```

On the second view controller, we have `MainClass` object as `main` parameter. Here is the code for SecondViewController.

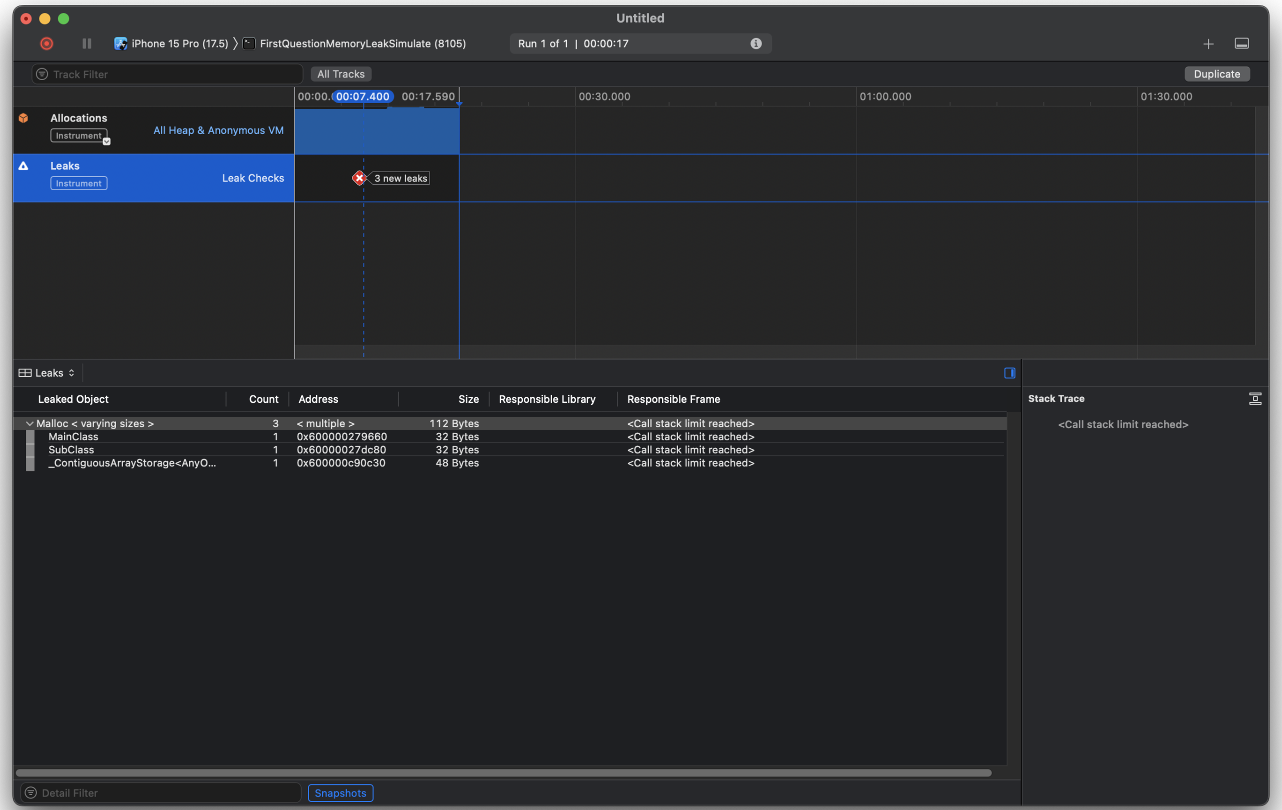
```
class SecondViewController: UIViewController {

    let main = MainClass()

    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = .red
        main.add(sub: SubClass(main: main))
    }

    deinit {
        print("\(Self.self) object was deallocated")
    }
}
```

Now lets check on Leaks Instrument. If we navigate to the second view controller and dismiss it back to the first view controller, we can see the leaks intrument detects a memory leak. If we select the memory leak icon, it will give us the object that causing the leak.



Leaked Object	Count	Address	Size	Responsible Library	Responsible Frame
Malloc < varying sizes >	3	< multiple >	112 Bytes		<Call stack limit reached>
MainClass	1	0x600000279660	32 Bytes		<Call stack limit reached>
SubClass	1	0x60000027dc80	32 Bytes		<Call stack limit reached>
_ContiguousArrayStorage<AnyO...	1	0x600000c90c30	48 Bytes		<Call stack limit reached>

Stack Trace

<Call stack limit reached>

As you can see, There's a leak on MainClass and SubClass class. From the timestamp and action we did, we know that leak happens when we dismiss second view controller.

When we checked the codebase, we saw the strong reference on `MainClass` parameter causing the class cannot be fully destroyed when the view is dismissed. We can fix it by changing it to weak var. We put a deinitializer to trigger a print action when the class is destroyed.

```
class SubClass {
    weak var main: MainClass?

    init (main: MainClass) {
        self.main = main
        self.main?.add(sub: self)
    }

    deinit {
        print("\(Self.self) object was deallocated")
    }
}
```

So you can see after we change it to weak var, the class used on the second view controller is destroyed when the second view controller is dismissed, solving our memory leak issue

2

2A. How can we manage tasks in a multithreaded environment? Give us an example on how to use the Grand Central Dispatch properly to divide tasks in different threads.

Multithreading is so important to manage load and tasks. If we do all tasks on the main thread, it will bog down the main thread and make the user experience bad. We can manage tasks using Grand Central Dispatch. We use `DispatchQueue` code block to send tasks to either the main thread or the background thread, and we can give labels and priority to the task. For example:

```swift
DispatchQueue.main.async {
    updateValueOnUI()
}
```

With `DispatchQueue.main`, we make sure the app do `updateValueOnUI()` tasks on main thread. The best practices is to only use main thread for UI.

For heavy tasks, we delegate tasks to the background with `DispatchQueue.global()` code written inside the code block will go to background thread.

```swift
DispatchQueue.global().async {
    heavyTask()
}
```

We can also set priority on it with QoS parameter like `.background`, `.default`, `.unspecified`, `.userInitiated`, `.userInteractive`, `.utility`. Here's how we can combine using background thread and main thread.

```swift
DispatchQueue.global(qos: .background).async {
    let taskResult = heavyTaskResults()

    DispatchQueue.main.async {
        updateUIWithValue(value: taskResult)
    }
}
```

in this block of code, we do heavy tasks on background. After we get results, we update the UI with the latest result on main thread.

2B. How can we manage tasks and wait for other tasks to finish before moving on to another task to start?

We can use grand central dispatch to manage sync and async task, or use async/await API.

For Grand Central Dispatch, there's a method for both sync and async. If we want the tasks to done synchronously, we can use this example:

```
doSomething()

DispatchQueue.global(qos: .background).sync {
    heavyTask()
}

afterTask()
```

In this block of code, `afterTask()` is triggered right after `heavyTask()`

We can also use `DispatchGroup()` to notify that all tasks is already complete, thus triggering the follow up task.

```
let dispatchGroup = DispatchGroup()

let queue = DispatchQueue.global(qos: .userInitiated)

dispatchGroup.enter()
queue.async {
    doTask1()
    dispatchGroup.leave()
}

dispatchGroup.enter()
queue.async {
    doTask2()
    dispatchGroup.leave()
}

dispatchGroup.notify(queue: DispatchQueue.main) {
    doTask3()
}
```

in this code, we setup a queue of tasks on global thread. We use `DispatchGroup()` to track whether the tasks is already complete or not. before the tasks, we trigger the `enter()` method. after the task is completed on background thread, we call `leave()` method. When all tasks already close, then the dispatchGroup will trigger the notify code block, and perform `doTask3()`. Remember all `enter()` method triggered must be closed with `leave()` method, otherwise the `.notify(queue: DispatchQueue.main)` code block will not be triggered.

We can also use `async/await` for it. its also similar, function with `async` will be done on background thread, and to trigger it we use `await` to wait for the tasks to complete before moving on to other code. We async method with `Task` on background thread. For example:

```
// example of async method
func method() async {
    await doHeavyTask()
}

// how to run async method
Task {
    before()
    await method()
    after()
}
```

2C. How do you perform multithreaded tasks in Core Data?

We can do use the `perform` and `performAndWait` method on `NSManagedContext` to perform multithreaded tasks in Core Data. all action inside the perform and performAndWait bracket will be done on separate thread, The difference between perform and performAndWait is the way in which the specified block of code is executed, asynchronously or synchronously.

We can also use async/await on swift, and specify the thread used on that task in Task block.

3. Please create a sample project that will draw/render these keypoints from the existing order from the JSON (2D graph with core graphics, 3D graph with SceneKit framework)

Here is the Github link : https://github.com/frhamadiansyah/IdeasLabTest

4. Is there any way in Swift to normalize the output of the rendered keypoints so it would always be the same size/scale?

We can do it by scale the keypoints and normalize each coordinate with the same scale.

- Find the min max value for each 3 axis on all data points
- Find the width, height, and depth of the dataset by subtract max value with min value for each respectable axis
- find the maximum value of the width, height, and depth. Use that as the scale. We only use 1 scale so we keep the same normalize distance between points
- Divide each coordinate x, y z value with the scale