7) Memory Management

1) What is Reference Counting and how does it work?
   In the Reference Counting mechanisms every object keeps a counter indicating the number of other objects that hold a reference to the object. The object is not deallocated unless the counter drops to zero. Thus, it is prone to cyclic references that prevent the participating objects from being freed.

2) When is an object retained and released?
   In Swift, an object is retained when it is assigned to a variable or property (default behavior) and released as soon as the scope of the variable or property is left or if the reference is set to nil.

3) On which types is Reference Counting applied?
   Reference Types, e.g. Classes & Closures

4) Why are IBOutlets declared ‘weak’ by default when a connection between view controller and view is created via interface builder?
   The object instance to which the outlets variable points is retained by the object graph of the view or the runtime system and does not need to be retained by the view controller.

5) Explain the difference between weak and unowned properties and the implication on the type of the property declared as either weak or unowned.
   Weak properties are allowed to become nil and must be declared as a variable whereas unowned properties cannot become nil within their lifetime and may also be constants.

4) Does the following code snippet contains a potential retain cycle? (Y/N) – Y

```swift
class Car {
    var owner: CarDriver
}

class Person {
    let firstName, lastName: String
}

class CarDriver: Person {
    var car: Car?
}
```

1) If the code snippet contains a potential retain cycle, explain where the cycle could occur and how you would resolve the issue. If no retain cycle is present, explain why there is no retain cycle between the instances of the classes.
   The cycle would occur between CarDriver.car and car.owner. Declaring CarDriver.car as weak would be one possible solution. (Choosen because the type vor car is already an optional)

5) Does the following code snippet contains a potential retain cycle? (Y/N) – N

```swift
class BooksViewController: UITableViewController {
    var books = [Book]()

    override func prepareForSegue(_ segue: UIStoryboardSegue, sender: AnyObject) {
        if segue.identifier == "AddBookSegue" {
            if let controller = segue.destinationViewController as? AddBookViewController {
                * controller.delegate = { [weak self] (book) -> Void in
                    if let b = book {
                        self?.books.append(b)
                    }
                }
            }
        }
    }
}
```
class AddBookViewController: UIViewController {
    var completionHandler: ((book: Book?) -> Void)?

    @IBAction save(sender: AnyObject) {
        let book = Book(/** Initialized from Outlets */)
        completionHandler?(book)
    }

    @IBAction cancel(sender: AnyObject) {
        completionHandler?(nil)
    }
}

1) If the code snippet contains a potential retain cycle, explain where the cycle could occur and how you would resolve the issue. If no retain cycle is present, explain why there is no retain cycle between the instances of the classes.

   A retain cycle in this code example can never occur for two reasons: First, there is no strong reference between BooksViewController and AddBookViewController at the scope of the class. The strong reference in the prepareForSegue(_:sender:) method is released as soon as the method’s scope is left. Second, the self-reference in the closure used for the delegate property of AddBookViewController is implemented as a weak reference, which also prevents any possible retain cycles.