Chapter 4: Enumerations and Pattern Matching

```swift
enum Dimension {
    case us(Double, Double)
    case metric(Double, Double)
}

func convert(dimension: Dimension) -> Dimension {
    switch dimension {
    case .us(let length, let width):
        return .metric(length * 0.304, width * 0.304)
    }
}

let convertedDimension = convert(dimension: Dimension.metric(5.0, 4.0))
print(convertedDimension)

let anyValue: Any = 7

switch anyValue {
    case is Int: print(anyValue + 3)
    case let ourValue as Int: print(ourValue + 3)
    default: ()
}
```

Note: The code snippet includes a comment indicating that the switch must be exhaustive, considering adding a default clause.
Chapter 5: Generics and Associated Type Protocols

```swift
var name = "John Doe"
var phoneNumber = 5141111111

let (a, b) = swapTwoValues(a: name, b: phoneNumber)

func format(a: String) -> String {
    return "formatted \(a)"
}

func appendStrings(a: String, b: String) -> String {
    return a + b
}

print("The result is: \(calculate(a: "2", b: "2", funcA: appendStrings, funcB: format))")
```

(Errors indicated by comments)
Chapter 7: Dealing with Optionals

```swift
var aString: String = "A String literal"
aString = nil // Compile error

let optionalDict: Dictionary<String, Int>? = ["One": 1, "Two": 2, "Three": 3]
let implicitlyUnwrappedDict: Dictionary<String, Int>! = ["One": 1, "Two": 2, "Three": 3]

let firstValue = optionalDict["One"] // Value of optional type 'Dictionary<String, Int>?' not unwrapped, did you mean to use '?' or '??'
let implicitlyUnwrappedFirstValue = implicitlyUnwrappedDict["One"]
```
Chapter 8: Functional Data Structures
Chapter 11: Case Study - Developing an iOS Application with FP and OOP Paradigms
### Choose a template for your new project:

<table>
<thead>
<tr>
<th>iOS</th>
<th>Application</th>
<th>Framework &amp; Library</th>
<th>watchOS</th>
<th>Application</th>
<th>Framework &amp; Library</th>
<th>tvOS</th>
<th>Application</th>
<th>Framework &amp; Library</th>
<th>OS X</th>
<th>Application</th>
<th>Framework &amp; Library</th>
<th>System Plug-in</th>
<th>Other</th>
</tr>
</thead>
</table>

#### Single View Application

This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a storyboard or nib file that contains the view.