

POGIL Activity 1.1: Introduction to Swift

In this activity, we will learn about the Swift and how to investigate features of a programming language that help you choose what is appropriate for a given problem.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: Team Get Swifty

Date: 8/25/2020

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Brian Lucero
Presenter. Talks to the facilitator and other teams.	Fahad Alsowaylim
Reflector. Considers how the team could work and learn more effectively.	Brandon Ryan
Recorder. Records all answers and questions and makes the necessary submission.	Angel Zambrano



Part 1. Explore (10 min)

Start time: 2:33

1. Recall your experiences with programming in C++. Identify at least three strengths and weaknesses of the language. If you did not take a C++ course, you can choose another programming language that you used instead.

Strengths:

1. Analytical Thinking
2. Efficient choice of Data Structures
3. Similarity to other popular languages

Weaknesses:

1. Memory Management
2. Low level syntax
3. Game dev (GUI)

Part 2. Invent (5 min)

Start time: _____

2. Why did Apple choose to create the new programming language, Swift, instead of continuing to use Objective-C?

Basically, you write to your computer as if it were a person(easier to read and write), it's safer and easy to maintain. Because, lazy, and encourages more people to work on iphone and macs.

Part 3. Apply (10 min)

Start time: _____

3. As a software developer when and why do you think you will use Swift over C++ (or the other programming language you selected in #1)?

iOS app development



Reflector questions

1. What was the most useful thing you learned during this session?

The members names and who will be doing what roles in the group

2. What did the team do well?

Handle issues that arose with document access for one of the members

3. What were the challenges that the team encountered?

One member didn't have access to the doc, and instead we shared the screen so he could participate before resolving the issue

4. What do you suggest the team do in the next meeting to do better?

Distribute workload sooner so that we are not on a time crunch

5. Rate your team according to the rubric below: 10

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0



POGIL Activity 1.2: Constants, Variables, and Types

In this activity, you will learn about Swift constants, variables, and types. You will be using them to create basic Swift programs.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: Team Get Swifty

Date: 8/27/2020

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Fahad Alsowaylim
Presenter. Talks to the facilitator and other teams.	Brian lucero
Reflector. Considers how the team could work and learn more effectively.	Angel Zambrano
Recorder. Records all answers and questions and makes the necessary submission.	Brandon Ryan



Part 1. Explore (10 min)

Start time: 1:40__

1. Analyze the codes below. The code on the first row is C++ code, while the code on the second row is Swift code. Both programs behave the same way.

C++ code

```
#include <iostream>

struct Person {
    const std::string firstName;
    const std::string lastName;

    void sayHello() {
        std::cout << "Hello there! My name is " << firstName
                  << " " << lastName << "." << std::endl;
    }
};

int main() {
    struct Person aPerson = {"Jacob", "Edwards"};
    struct Person anotherPerson = {"Candace", "Salinas"};
    aPerson.sayHello();
    anotherPerson.sayHello();
}
```

Swift code

```
struct Person {
    let firstName: String
    let lastName: String

    func sayHello() {
        print("Hello there! My name is \(firstName) \(lastName).")
    }
}

var shape = "Triangle";
let color = "Red";
var age = 20;

let aPerson = Person(firstName: "Jacob", lastName: "Edwards")
let anotherPerson = Person(firstName: "Candace", lastName:
"Salinas")

aPerson.sayHello()
```



```
anotherPerson.sayHello()
```

Excerpt From: Apple Education. "App Development with Swift." Apple Inc. - Education, 2019. Apple Books. <https://books.apple.com/us/book/app-development-with-swift/id1465002990>

Part 2. Invent (10 min)

Start time: __1:41__

2. What are the similarities and differences between the C++ and Swift code?

Similarities: struct, curly brackets for functions, print functions,

Differences: func to declare a function in swift, no main function

Part 3. Apply (10 min)

Start time: __1:47__

3. Create a structure in Swift that represents a restaurant. A restaurant should have a name and address. A restaurant's name will not change, but it could move to another address. Create a method that displays a message that will welcome it's customers to the restaurant. Preferably, the program should tell customers its name and address.

Create code that will create a restaurant. Feel free to use any name and address. Call the appropriate method to display its welcome message.

```
struct Restaurant {  
    let restName: String  
    var address: String  
    func welcome() {  
        print("Welcome to \(restName). We are located at  
        \(address).")  
    }  
}
```



```
let aRestaurant = Restaurant(restName: "Shoney's", address:
"711 Swifty blvd.")
aRestaurant.welcome()
```

Part 4. Extra challenge (5 min)

Start time: __1:57__

4. Follow the steps below depending on whether you have Xcode on your machine or not

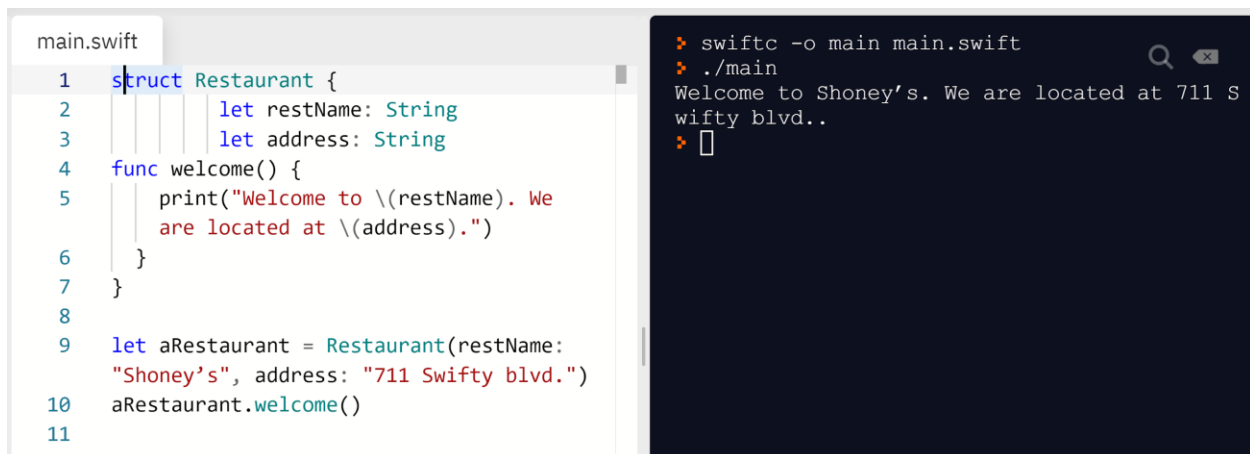
Xcode

- Select "Get started with a playground"
- Create a "Blank" playground and provide any name you see fit.
- Copy your answer in #3 into the editor and click play.

No Xcode

- Use an online compiler that supports Swift. For example, repl.it or [CS50 sandbox](https://cs50sandbox.com). You may need to create an account to access these websites.
- Copy your answer in #3 into the editor.

Each member of the group should take a screenshot of their Xcode playground or online compiler showing that the program worked and its output. Place each member's screenshot below.



The screenshot displays an Xcode playground window. On the left, the Swift code in `main.swift` is shown with line numbers 1 through 11. The code defines a `Restaurant` struct with `restName` and `address` properties, a `welcome()` function that prints a message, and an instance `aRestaurant` of the `Restaurant` struct. On the right, the terminal output shows the command `swiftc -o main main.swift` being executed, followed by the command `./main` which produces the output: `Welcome to Shoney's. We are located at 711 Swifty blvd..`

<pre>main.swift 1 struct Restaurant { 2 let restName: String 3 let address: String 4 func welcome() { 5 print("Welcome to \(restName). We are located at \(address).") 6 } 7 } 8 9 let aRestaurant = Restaurant (restName: "Chilies", address: "1634 Partition Way") 10 aRestaurant.welcome()</pre>	<pre>> swiftc -o main main.swift > ./main Welcome to Chilies. We are located at 1634 Partition Way.</pre>
<pre>main.swift 1 struct Restaurant { 2 let restName: String 3 let address: String 4 func welcome() { 5 print("Welcome to \(restName). We are located at \(address).") 6 } 7 } 8 9 let aRestaurant = Restaurant(restName: "Roberto's", address: "311 Park Avenue") 10 aRestaurant.welcome()</pre>	<pre>> swiftc -o main main.swift > ./main Welcome to Roberto's. We are located at 311 Park Avenue.</pre>
<pre>main.swift 1 struct Restaurant { 2 let restName: String 3 let address: String 4 func welcome() { 5 print("Welcome to \(restName). We are located at \(address).") 6 } 7 } 8 9 10 let aRestaurant = Restaurant(restName: "Shoney's", address: "711 Swifty blvd.") 11 aRestaurant.welcome() 12</pre>	<pre>> swiftc -o main main.swift > ./main Welcome to Shoney's. We are located at 711 Swifty blvd..</pre>
<pre>main.swift 1 struct Restaurant { 2 let restName: String 3 let address: String 4 func welcome() { 5 print("Welcome to \(restName). We are located at \(address).") 6 } 7 } 8 9 10 let aRestaurant = Restaurant(restName: "Shoney's", address: "HW1 Sheikh Khalifa rd.") 11 aRestaurant.welcome() 12</pre>	<pre>> swiftc -o main main.swift > ./main Welcome to Shoney's. We are located at HW1 Sheikh Khalifa rd..</pre>

Reflector questions

1. What was the most useful thing you learned during this session?

How the Swift Language works and the differences/similarities it had with c++.

2. What did the team do well?

The Team was able to work effectively and efficiently on the problems at hand.

3. What were the challenges that the team encountered?

Trying to figure out how the two languages were similar.

4. What do you suggest the team do in the next meeting to do better?

Nothing really, perhaps deciding our roles quicker.

5. Rate your team according to the rubric below:

10

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0



POGIL Activity 1.3: Operators and Conditional Statements

In this activity we will learn about Swift operators and conditional statements.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: Team Get Swifty

Date: 9/1/2020

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Fahad Alsowaylim
Presenter. Talks to the facilitator and other teams.	Brian lucero
Reflector. Considers how the team could work and learn more effectively.	Angel Zambrano
Recorder. Records all answers and questions and makes the necessary submission.	Brandon Ryan



POGIL Template by Paul Salvador Inventado is licensed under a Creative Commons

Attribution-NonCommercial-ShareAlike 4.0 International License. Based on a work at «<http://bit.ly/2LtrlKv>».

Part 1. Explore (10 min)

Start time: _____

For the following activities, you can consult your AWD or SPL books, or search for information over the internet.

1. What operators are available in the Swift programming language? No need to include nil-coalescing operators.

```
+, -, *, /, %, =, ==, &&, !a (the not operator is part of the  
logical operators group),
```

2. What are the different Swift conditional statements?

```
For-in, While, Repeat-While, (if, else if, else ), Guard, Switch
```

Part 2. Invent (30 min)

Start time: _____

3. Give one or more examples that show the if, else if, and else statements in Swift. How are they similar or different from other programming languages that you used?

Examples:

```
if firstName == "Tod" {  
    print("Hey Tod")  
} else if firstName == "Richard" {  
    print("Hey Richard")  
} else {  
    print("What's your name?")  
}
```

Similarities: The conditional statements are similar in the structure to C++ and the conditions are in the same place.

Differences: It is different from python since that the else if statement is not written as elif in python.



4. Swift's switch statement supports several types of cases. Give an **example** for the two of the four types of cases and **explain how it works**. Your group will be assigned either of the two sets (1) interval matching and where; or (2) tuples and value binding. You can find this in the SPL book under the "Control Flow" topic, but you are free to consult the internet for other resources.

Case type 1: Interval Matching

Example:

```
let wavelength = 620

switch wavelength {
case 380..<450:
    print("Purple!")
case 450..<495:
    print("Blue!")
case 495..<570:
    print("Green!")
case 570..<590:
    print("Yellow!")
case 590..<620:
    print("Orange!")
case 620..<750:
    print("Red!")
default:
    print("Not in visible spectrum")
}
```

Usage:

Used for if a certain value falls within a continuous range of values instead of a single discrete value

Case type 2: Where

Example:

```
let yetAnotherPoint = (1, -1)
switch yetAnotherPoint {
case let (x, y) where x == y:
    print("\(x), \(y) is on the line x == y")
case let (x, y) where x == -y:
    print("\(x), \(y) is on the line x == -y")
case let (x, y):
    print("\(x), \(y) is just some arbitrary point")
}
```

Usage:

A *where* case can be used with a where clause to check for additional conditions.

Part 3. Apply (20 min)

Start time: _____

5. Create a Swift program according to the specifications below. Write your program in the box provided.
- Ask the user to provide the number of anime series they've already watched
 - Ask the user to provide the title of their favorite anime
 - If the user's favorite anime happens to be Dragon Ball or Gundam, your program should display "Gotta love the classics!"
 - Your program should also display the following messages based on the number of anime series the user provided
 - Less than 5: You need to watch more anime!
 - Between 5 and 10 (inclusive): Way to go!
 - Between 11 and 20 (inclusive): Anime lover!
 - Over 20: すばらしい
(read as subarashii - it is Japanese for splendid; glorious; excellent; or superb)

Swift code:

```
print ("How many anime have you watched?")
let numberA = Int(readLine()!)!

print("What is the title of your favorite anime?")
let title = readLine()

// if title == "Dragon Ball" || title == "Gundam" {
// print("Gotta love the classics!")
// }

switch title {
case "Dragon Ball", "Gundam":
    print("Gotta love the classics!")
default:
    print()
}

switch numberA {
case 0...4:
    print("You need to watch more Anime!")
case 5...10:
    print("Way to go!")
```



```

case 11...20:
    print("Anime lover!")
case 21...Int.max:
    print("すばらしい")
default:
    print("You owe it to yourself to watch more anime!!")
}

```

6. Write your Swift program and run it using any platform you choose (e.g., Swift playground, repl.it, [CS50 sandbox](https://cs50sandbox.com)). Take a screenshot of the output and place it in the box below.

The screenshot shows a Swift playground with a file named 'main.swift'. The code in the file is as follows:

```

1  print ("How many anime have you watched?")
2  let numberA = Int(readLine()!)!
3  //-----
4  print("What is the title of your favorite anime?")
5  let title = readLine()
6  //-----
7  if title == "Dragon Ball" || title == "Gundam" {
8  print("Gotta love the classics!")
9  }
10 //-----
11 switch numberA {
12 case 0...4:
13 | print("You need to watch more Anime!")
14 case 5...10:
15 | print("Way to go!")
16 case 11...20:
17 | print("Anime lover!")
18 case 21...Int.max:
19 | print("すばらしい")
20 default:
21 | print("You owe it to yourself to watch more anime!!")
22 }
23

```

The output on the right shows the program's execution:

```

> swiftc -o main main.swift
> ./main
How many anime have you watched?
13
What is the title of your favorite anime?
Dragon Ball
Gotta love the classics!
Anime lover!
>

```

Reflector questions

1. What was the most useful thing you learned during this session?

All the features for the swift language

2. What did the team do well?

The Team was able to work together well

3. What were the challenges that the team encountered?

Trying to figure out what some questions where asking

4. What do you suggest the team do in the next meeting to do better?

Bring in the Instructor sooner when a question arises.

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0

POGIL Activity 2.1: Strings and Functions

In this activity we will learn about strings and functions in Swift

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: **Team Get Swifty**

Date: **9/1/2020**

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Brandon Ryan
Presenter. Talks to the facilitator and other teams.	Angel Zambrano
Reflector. Considers how the team could work and learn more effectively.	Fahad Alsowaylim
Recorder. Records all answers and questions and makes the necessary	Brian Lucero



POGIL Template by Paul Salvador Inventado is licensed under a Creative Commons

Attribution-NonCommercial-ShareAlike 4.0 International License. Based on a work at «<http://bit.ly/2LtrlKv>».

submission.	
-------------	--



Part 1. Explore (5 min)

Start time: 2:07pm

1. What will be the screen output of the code below?

```
func makePizza(sauce: String,
               topping1: String,
               topping2: String,
               topping3: String) -> String {
    var pizza = ""
    pizza = pizza + "pizza with " + sauce + " sauce"
    pizza = pizza + ", " + topping2
    pizza = pizza + " and " + topping3
    pizza = "Freshly baked " + pizza
    return pizza
}

let my_awesome_pizza = makePizza(sauce: "tomato", topping1:
                                "pepperoni", topping2: "Ham", topping3:
                                "cheese")

print("I'd like to order a \(my_awesome_pizza)")
```

Output: I'd like to order a freshly baked pizza with tomato sauce, Ham and cheese

Part 2. Invent (25 min)

Start time: __2:13__

For the following questions, you can consult your AWD or SPL books, or search for information over the internet.

2. Describe the concepts listed below and copy code from #1 that shows the concept.

String concatenation: string concatenation is the operation of joining character strings end-to-end. For example, the concatenation of "snow" and "ball" is "snowball".
String concatenation is as simple as combining two strings with the + operator
Sample code from #1:

Ex: a)

```
sauce: String,
topping1: String,
topping2: String,
topping3: String) -> String
```

Ex: b)



POGIL Template by Paul Salvador Inventado is licensed under a Creative Commons

Attribution-NonCommercial-ShareAlike 4.0 International License. Based on a work at «<http://bit.ly/2LtrlKv>».

18 of

```
pizza + "pizza with " + sauce + " sauce"
```

```
pizza = pizza + "pizza with "
```

String interpolation:

String interpolation is a way to construct a new String value from a mix of constants, variables, literals, and expressions by including their values inside a string literal.

Sample code from #1:

```
pizza = pizza + ", " + topping2
```

Note: topping2 gets replaced with “Ham”

3. Describe the parts of the makePizza function.

```
makePizza
```

Description: A function that takes in 4 parameters and returns a string variable “pizza”, which contains a group of strings that have been concatenated together to form the one string pizza

```
(sauce: String, topping1: String, topping2: String, topping3: String)
```

Description: the 4 string variable parameters that eventually compose the string variable pizza by string concatenation

```
-> String
```

Description: The arrow indicates the return type for the function makePizza

4. What are argument labels in functions? Provide an example

Argument labels are what you pass to the function to give it a more sentence like structure.

EX: `makePizza(sauce: "tomato", topping1: "pepperoni", topping2: "Ham", topping3: "cheese")`

5. How would you omit argument labels in functions? Provide an example of the function and a call to that function.

Write an underscore, instead of an explicit argument label for that parameter

Part 3. Apply (10 min)

Start time: ____2:29____

6. Create a function that predicts the type of fruit given its weight and color. It should return one of the following: apple, orange, watermelon, or unknown. Use the table below to guide the design of your solution.

Fruit	Color	Weight rangeWeight (oz.)
apple	red	4 - 8
orange	orange	3 - 7
watermelon	green	320 - 400

```
func fruitChecker(weight: Int, color: String) -> String {
    var fruit = ""
    if (4 < weight && weight < 8 && color == "red") {
        fruit = "apple"
    } else if (3 < weight && weight < 7 && color == "orange")
        fruit = "orange"
    } else if (320 < weight && weight < 400 && color == "green")
        fruit = "watermelon"
    } else {
        fruit = "unknown"
    }
    return fruit
}
```

Reflector questions

1. What was the most useful thing you learned during this session?

Argument labels

2. What did the team do well?

We were open to each other's suggestions

3. What were the challenges that the team encountered?

Not enough time for the coding part

4. What do you suggest the team do in the next meeting to do better?

Do the code first??

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0



POGIL Activity 2.2: Structures

In this activity we will learn about structures and how to use them to make code more reusable.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____ **Team Get Swifty** _____ Date: __9/3/2020__

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Angel Zambrano
Presenter. Talks to the facilitator and other teams.	Fahad Alsowaylim
Reflector. Considers how the team could work and learn more effectively.	Brian Lucero
Recorder. Records all answers and questions and makes the necessary submission.	Brian Lucero



Part 1. Explore (15 min)

Start time: __1:40__

1. Consider the code below. What are the potential issues with the code design? For example, how will you change the code if you want to cook five steaks?

```
let uncookedSteakWeight = 8.0
var internalSteakTemperature = 100.0
var doneness = "Uncooked"
var cookedSteakWeight: Double

print("Internal temperature: \(internalSteakTemperature)°F\n")
print("Doneness: \(doneness)\n")
print("Weight: \(uncookedSteakWeight) oz.\n")

while (doneness != "Medium well") {
    internalSteakTemperature += 5
    switch internalSteakTemperature {
        case 0..<125:
            doneness = "Uncooked"
            cookedSteakWeight = uncookedSteakWeight
        case 125..<135:
            doneness = "Rare"
            cookedSteakWeight = uncookedSteakWeight * 0.95
        case 135..<145:
            doneness = "Medium rare"
            cookedSteakWeight = uncookedSteakWeight * 0.90
        case 145..<150:
            doneness = "Medium"
            cookedSteakWeight = uncookedSteakWeight * 0.85
        case 150..<160:
            doneness = "Medium well"
            cookedSteakWeight = uncookedSteakWeight * 0.80
        case 160..<170:
            doneness = "Well done"
            cookedSteakWeight = uncookedSteakWeight * 0.75
        default:
            doneness = "Burnt"
            cookedSteakWeight = uncookedSteakWeight * 0.70
    }
    print("Internal temperature: "
        + "\(internalSteakTemperature)°F\n")

    print("Doneness: \(doneness)\n")
    print("Weight: \(cookedSteakWeight) oz.\n")
}
```



Potential issues: if you wanted to cook five steaks, you would have to run the program 5 different times. The program only accepts information for it to cook a single steak and then ends. Instead, maybe we could use the parts of this program and create a `steakcooker()` function instead.

2. Read Unit 2.3 in the ADS book on Structures. How do you think structures can be used to resolve issues you identified in #1?

We can utilize various program structure concepts to resolve the program having to be ran multiple times. For example, running the program in a loop and would repeat the number of times desired.

Part 2. Invent (15 min)

Start time: __1:50__

3. What is a computed property? Give an example.

Description: whereas computed properties calculate (rather than store) a value. - (swift.org)

Example:

```
9  var x = 10
10 var y = 20
11
12 var extProperty: Int {
13     get { // we can simplify it and remove get
14         return x+y
15     }
16 }
17
18 print(extProperty) //30
19
```

-google

4. What is a property observer? Give an example.

Description: Property observers observe and respond to changes in a properties value, they are called every time a properties value is set. -(swift.org)

Example:

- a) If the internalSteakTemperature variable were in a struct and it was changing, then the property observer would observe it since it is a stored property.

b)

```
class StepCounter {
    var totalSteps: Int = 0 {
        willSet(newTotalSteps) {
            print("About to set totalSteps to \(newTotalSteps)")
        }
        didSet {
            if totalSteps > oldValue {
                print("Added \(totalSteps - oldValue) steps")
            }
        }
    }
}

let stepCounter = StepCounter()
stepCounter.totalSteps = 200
// About to set totalSteps to 200
// Added 200 steps
stepCounter.totalSteps = 360
// About to set totalSteps to 360
// Added 160 steps
stepCounter.totalSteps = 896
// About to set totalSteps to 896
// Added 536 steps
```

The StepCounter class declares a totalSteps property of type Int. This is a stored property with willSet and didSet observers.

The willSet and didSet observers for totalSteps are called whenever the property is assigned a new value. This is true even if the new value is the same as the current value.

This example's willSet observer uses a custom parameter name of newTotalSteps for the upcoming new value. In this example, it simply prints out the value that is about to be set.

The didSet observer is called after the value of totalSteps is updated. It compares the new value of totalSteps against the old value. If the total number of steps has increased, a message is printed to indicate how many new steps have been taken. The didSet observer does not provide a custom parameter name for the old value, and the default name of oldValue is used instead.

c)

Part 3. Apply (25 min)

Start time: __ 1:57 __

5. Refactor the code in #1 using structures. Make sure you use either **computed properties** or **property observers** in your solution. The solution will contain a structure and the loop that cooks the steak until it is Medium well. Hint: You will need to understand the following concepts to solve the problem: structures, structure instances, initializers, instance methods, computed properties, and property observers.

```

struct Steak {

    let uncookedSteakWeight = 8.0
    var internalSteakTemperature = 100.0
    var doneness = "Uncooked"
    var cookedSteakWeight: Double

    func printInternalTemp() {
        print("Internal temperature: \(internalSteakTemperature)°F\n")
    }

    func printDoneness() {
        print("Doneness: \(doneness)\n")
    }

    func printWeight() {
        print("Weight: \(uncookedSteakWeight) oz.\n")
    }

    func cook() {
        while (doneness != "Medium well") {
            internalSteakTemperature += 5
            switch internalSteakTemperature {
                case 0..<125:
                    doneness = "Uncooked"
                    cookedSteakWeight = uncookedSteakWeight

```



```

        case 125..<135:
            doneness = "Rare"
            cookedSteakWeight = uncookedSteakWeight * 0.95
        case 135..<145:
            doneness = "Medium rare"
            cookedSteakWeight = uncookedSteakWeight * 0.90
        case 145..<150:
            doneness = "Medium"
            cookedSteakWeight = uncookedSteakWeight * 0.85
        case 150..<160:
            doneness = "Medium well"
            cookedSteakWeight = uncookedSteakWeight * 0.80
        case 160..<170:
            doneness = "Well done"
            cookedSteakWeight = uncookedSteakWeight * 0.75
        default:
            doneness = "Burnt"
            cookedSteakWeight = uncookedSteakWeight * 0.70
    }
} // end of struct

let steakA = Steak(uncookedSteakWeight: 8.0,
internalSteakTemperature: 100.0, doneness: "Uncooked",
cookedSteakWeight: Double)

// steak properties before cooking:
steakA.printInternalTemp()
steakA.printDoneness()
steakA.printWeight()

// steak gets cooked:
steakA.cook()

// steak properties after being cooked:
steakA.printInternalTemp()
steakA.printDoneness()
steakA.printWeight()

```

Reflector questions

1. What was the most useful thing you learned during this session?

Learning to refactor code in a program in to structures to be reused during run time w/o having to re run



2. What did the team do well?

Collaborated easily amongst each other to determine acceptable solutions

3. What were the challenges that the team encountered?

Everyone being able to access the ADS book, listen more intently for instructions on where to stop within the activity; completed the whole activity then realized went too far ahead afterwards.

4. What do you suggest the team do in the next meeting to do better?

Prepare a little bit beforehand to have resources available

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0

POGIL 3.1 on next page --->



POGIL Template by Paul Salvador Inventado is licensed under a Creative Commons

Attribution-NonCommercial-ShareAlike 4.0 International License. Based on a work at «<http://bit.ly/2LtrlKv>».

POGIL Activity 3.1: Managing Structures

In this class activity we will learn about initializing structures and creating methods that manipulate them.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: **Team Get Swifty** Date: **9/8/2020**



POGIL Template by Paul Salvador Inventado is licensed under a Creative Commons

Attribution-NonCommercial-ShareAlike 4.0 International License. Based on a work at «<http://bit.ly/2LtrlKv>».

29 of

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Angel Zambrano
Presenter. Talks to the facilitator and other teams.	Brian Lucero
Reflector. Considers how the team could work and learn more effectively.	Angel Zambrano
Recorder. Records all answers and questions and makes the necessary submission.	Fahad Alsowaylim



Part 1. Explore (10 min)

Start time: 1:49

1. Imagine an oven. Make a list of properties that we can use to describe it as well as store information to perform its functions? Make another list of an oven's functions.

Oven
Properties: Temperature, Timer, Storage Capacity, Cook Setting, Light switch (if door is open/not open)
Functions: setTemp(), cookSetting(), doorOpen(bool).

2. What should we use as the initial values of the oven's properties?

Temperature: 70 degrees (room temp)
 Timer: 0
 Storage Capacity: 30in
 Cook Setting: none
 Light switch: off

3. Which of the oven's functions will change one or more of its properties?

All?

Part 2. Invent (10 min)

Start time: 1:56

4. Read on structure initializers in the ADS book. How can you use initializers to set the default values of the oven's properties? Create an initializer for your Oven structure. Provide the code below.



1. Assign the values directly
2. `init() { default values }`
 - a. Similar to default constructor from c++
3. `init(values of your choice) { default values corresponding to values }`
 - a. Similar to non-default constructor from c++

```
struct Oven {  
    var temperature: Double  
    var timer: double  
    var storageCapacity: Double  
    var cookSetting: String  
    var lightSwitch: boolean  
  
    init( ) {  
        temperature = 70  
        timer = 0  
        storageCapacity = 30  
        cookSetting = "none"  
        lightSwitch = false  
    }  
}
```

5. What keyword does Swift require so you are able to create a method that modifies a structure's properties?

Modifying Value Types from Within Instance Methods

Structures and enumerations are *value types*. By default, the properties of a value type cannot be modified from within its instance methods.

However, if you need to modify the properties of your structure or enumeration within a particular method, you can opt in to *mutating* behavior for that method. The method can then mutate (that is, change) its properties from within the method, and any changes that it makes are written back to the original structure when the method ends. The method can also assign a completely new instance to its implicit `self` property, and this new instance will replace the existing one when the method ends.

You can opt in to this behavior by placing the `mutating` keyword before the `func` keyword for that method:

```

1  struct Point {
2      var x = 0.0, y = 0.0
3      mutating func moveBy(x deltaX: Double, y deltaY: Double) {
4          x += deltaX
5          y += deltaY
6      }
7  }
8  var somePoint = Point(x: 1.0, y: 1.0)
9  somePoint.moveBy(x: 2.0, y: 3.0)
10 print("The point is now at \(somePoint.x), \(somePoint.y)")
11 // Prints "The point is now at (3.0, 4.0)"

```

The key word would be “mutating”

Part 3. Apply (10 min)

Start time: 2:10

- Implement the oven structure you designed in #1, but assume that the oven’s controls are mechanical. That means you can only increase or decrease values and not change them directly (e.g., rotate the dial to incrementally increase the temperature). Make sure to provide properties, instance methods, and initializers.



POGIL Template by Paul Salvador Inventado is licensed under a Creative Commons

Attribution-NonCommercial-ShareAlike 4.0 International License. Based on a work at «<http://bit.ly/2LtrlKv>».

33 of

```
struct Oven {
    var temperature: Double
    var timer: double
    var storageCapacity: Double
    var cookSetting: String
    var lightSwitch: boolean

    init( ) {
        desiredTemp = 0.0
        temperature = 70.00
        timer = 0.00
        storageCapacity = 30
        cookSetting = "none"
        lightSwitch = false
    }

    mutating func setTemp( direction: Boolean){

        if(direction)
            Temperature += 1
            print(temperature)
        else
            Temperature -= 1
            print(temperature)

    }

    func cookSetting(cook: String){
        cookSetting = cook
    }

    mutating func doorOpen(lightswitch door: bool) {

        door = true ? : lightSwitch = true : lightSwitch = false

    }
} // end of struct
```

Reflector questions

1. What was the most useful thing you learned during this session?

We learned about what goes into making a struct and what our thought process should be.

2. What did the team do well?

We were able to understand most problems in an efficient manner and completed the problems swiftly.

3. What were the challenges that the team encountered?

A challenge that we encountered was not being able to understand the objective of a question at the end for a while.

4. What do you suggest the team do in the next meeting to do better?

Read the question more carefully and call the instructor when a serious question arises.

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
----------	-------



Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0



POGIL Activity 3.2: Classes, Objects, and Inheritance

In this activity you will learn about classes in Swift. Classes are very similar to structures, but they allow inheritance and are reference types. You will find out more about when to use structures and classes.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: **Team Get Swifty** Date: **9/10/2020**

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Brian Lucero
Presenter. Talks to the facilitator and other teams.	Angel Zambrano
Reflector. Considers how the team could work and learn more effectively.	Fahad Alsowaylim
Recorder. Records all answers and questions and makes the necessary submission.	Brian Lucero



Part 1. Explore (15 min)

Start time: _1:33_

1. Analyze the codes below and observe the difference between structures and classes.

```
// Structures
struct SPoint {
    var x: Int
    var y: Int
}

var mySPoint = SPoint(x: 10, y: 5)
var myOtherSPoint = mySPoint

print("Point: \(mySPoint.x)\n")
print("Other point: \(myOtherSPoint.x)\n")

myOtherSPoint.x = 20

print("Point: \(mySPoint.x)\n")
print("Other point: \(myOtherSPoint.x)\n")
```

Output:

Point: 10

OtherPoint: 10

Point: 10

OtherPoint: 20

```
// Classes
class CPoint {
    var x: Int
    var y: Int

    init(x:Int, y:Int) {
        self.x = x
        self.y = y
    }
}

var myCPoint = CPoint(x: 10, y: 5)
var myOtherCPoint = myCPoint

print("Point: \(myCPoint.x)\n")
print("Other point: \(myOtherCPoint.x)\n")

myOtherCPoint.x = 20
```



```
print("Point: \(myCPoint.x)\n")
print("Other point: \(myOtherCPoint.x)\n")
```

Output:

Point: 10

Other point: 10

Point: 20

Other point: 20

Differences between structures and classes:

- **Initializer function for class properties**
- **Copy assignment operator functionality for classes, changing property value for an object will change it for all other objects that were used to copy from**
- **Self.x - “self” is in reference to the class before any object has been initialized, used in setter or getter situations**
-

2. Consider the class diagrams below. What are the similarities and differences among the three types of Hikers?

NoviceHiker
totalSteps
walk stats

ExperiencedHiker
totalSteps totalClimbs
walk climb stats

ExpertHiker
totalSteps totalClimbs totalCamps
walk climb setupCamp stats

Similarities: They all have variables and functions
(with experience they add to them)

Differences: The more experienced Hikers have more variables and functions
(but still have the same variables/functions as the less experienced)

Part 2. Invent (15 min)

Start time: __2:08__

Read Unit 2.4 in the ADS book on Classes and Inheritance to answer the questions that follow.

3. Define inheritance and provide an example of how to implement inheritance in Swift.

Definition: Inheritance involves a Base class and corresponding Sub classes. Classes can call and access methods, properties, and subscripts belonging to their superclass and can provide their own overriding versions of those methods, properties, and subscripts to refine or modify their behavior.

Example:

```
class Subclass: BaseClass {  
    // subclass definition goes here  
}
```

4. What is method overriding and when should it be used?

Method overriding is when a method in a subclass has the same method name as superclass method name, but different implementation. Requires the override keyword to do this. So, instead of the subclass method performing implementation in superclass, it will perform its own tailored version. Subclass objects can use methods from superclass without having to re-define the method. So, if the class needs its own version of the same method, that's when override should be used.

Part 3. Apply (25 min)

Start time: __2:19__

5. Revisit the class diagram in #2. Which among the classes should be made into base classes and/or subclasses?



Base class: novice hiker
 Subclass 1: experienced hiker
 Subclass 2: expert hiker

6. Redesign the classes in #2 based on your answers in #5. Write the three classes in Swift to show the use of classes and inheritance. You can simply print out text descriptions for when methods are called.

```

Class noviceHiker
{
  Var totalSteps : Int

  init(ts: Int)
  {
    totalSteps = ts
  }

  func walk()
  {
    totalSteps++
  }

  func stats() -> Int
  {
    Return totalSteps
  }
} // end of noviceHiker class

Class experiencedHiker: noviceHiker {
  var totalClimbs: Int

  init( tc: Int ) {
    totalClimbs = tc
  }

  Func climb() {
    totalClimb++
  }

} // end of expHiker class

```

```
Class expertHiker: experiencedHiker
{
    Var totalCamps: Int

    init(tcamps: Int)
    {
        totalCamps = tcamps
    }

    func setUpCamp()
    {
        totalCamps++
    }
} //end of expertHiker class
```

7. When do you think it is best to use structures and when should you use classes?

Use classes when you have different types of objects for the same class, and you would like to use inheritance/override, and/or you want different objects to have their variables changed together..
Use Structs when you wanna keep it simple, easy, and don't need all the functionalities of a class

Reflector questions

1. What was the most useful thing you learned during this session?

Classes are reference types

2. What did the team do well?

Explaining to each other

3. What were the challenges that the team encountered?



none

4. What do you suggest the team do in the next meeting to do better?

Nothing

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0

POGIL Activity 4.1: Unit testing

In this activity we will learn the importance of testing code and the concepts of unit testing and test-driven development.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: Swifters _____

Date: ____ 9/15/20 ____



POGIL Template by Paul Salvador Inventado is licensed under a Creative Commons

Attribution-NonCommercial-ShareAlike 4.0 International License. Based on a work at «<http://bit.ly/2LtrlKv>».

43 of

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Scott Clary
Presenter. Talks to the facilitator and other teams.	Fahad Alsowaylim
Reflector. Considers how the team could work and learn more effectively.	T.J. Le
Recorder. Records all answers and questions and makes the necessary submission.	Hung Cun



Part 1. Explore (20 min)

Start time: _____

1. Design a palindrome function that takes in an integer and returns a boolean value. The function will check whether the given integer is a palindrome or not. A number is a palindrome if it reads the same backward or forward.

```
func palindrome (num: Int) -> Bool {  
    var reverse = 0  
    var digit = 0  
    var number = num  
    while (number != 0) {  
        digit = number % 10  
        reverse = (reverse * 10) + digit  
        number = number / 10  
    }  
    if (num == reverse) {  
        return true  
    } else {  
        return false  
    }  
}  
  
palindrome(num:1234321)
```

2. How confident are you about your answer? Why are you confident or not confident about your answer?

Yes...

3. Try the following values: 1111, 212, 1234321, 7887. Did your function give the correct answer for all cases?

Yes, it did.



4. You probably used one of three common solutions to the palindrome problem: (1) Use % and / to reverse the number then compare it with the original number; (2) Extract the first and last digit, compare them, and remove them until all numbers are checked or only one digit remains; and (3) Convert the number to a String, use the reverse function, and compare with the original value. Rewrite your palindrome function by using one of the implementations that you did not use in #1.

```
//let s = "hello"
// s[0..<3] "hel"
//s[3...] // "lo"

func Stringpalindrome (num: Int) -> Bool {
    let strnum = String(num)
    let revstr = String(strnum.reversed())
    if strnum == revstr {
        return true
    }else {
        return false
    }
}
Stringpalindrome(num: 12321)

//Experimental function//
func ispalindrome (integer: Int) ->Bool {

    var myString = String(integer)
    let length = myString.count
    if(myString.startIndex != myString.endIndex)
    {return false}
    if(length <= 2)
    {return true}
    else
    {return ispalindrome(integer: !Int(myString[1...length-2]))}
}
```

5. Try the following values: 1111, 212, 1234321, 7887. Did your function give the correct answer for all cases?

Yes

6. What were the advantages of having a set of values used for testing in relation to changing your function implementation?

They were small cases and made it easier to plan around

Part 2. Invent (15 min)

Start time: _____

Here is an example of a class that implements a unit test in Swift. It uses the XCTest library that provides several functions used for unit testing.

```
import XCTest           // Imports the XCTest Library.
@testable import Greetings // This imports the Greetings swift
                        // file in the project

// The class inherits from XCTestCase
class GreetingsTest: XCTestCase {

    override func setUpWithError() throws {
        // Put setup code here. This method is called before
        // the invocation of each test method in the class.
    }

    override func tearDownWithError() throws {
        // Put teardown code here. This method is called after
        // the invocation of each test method in the class.
    }

    // All test begin with the test keyword, followed by any
    // string

    func testEnglish() {
        let greeting: Greeting
        // Assertion that tests for equality
        XCTAssertEqual(greeting.english(), "hello")
    }
}
```

}

7. Go over [Apple's Developer Documentation](#), which is a good resource to find details about XCTest. You can see a list of all the different assertions that it supports under the Test Assertions section. Identify at least three other assertions, explain how it is used, and give an example. **Note: This testing framework only works with XCode.**

Assertion: Boolean**Description:** Test a condition that generates a true or false result.**Example:** `func XCTAssert(() -> Bool, () -> String, file: StaticString, line: UInt)`
Asserts that an expression is true.**Assertion: Error****Description:** Check whether a function call throws (or doesn't throw) an error.**Example:** `func XCTAssertThrowsError<T>(() -> T, () -> String, file: StaticString, line: UInt, (Error) -> Void)`
Asserts that an expression throws an error.**Assertion: Comparable Value****Description:** Compare two values to determine whether one is larger or smaller than the other.**Example:** `func XCTAssertGreaterThan<T>(() -> T, () -> T, () -> String, file: StaticString, line: UInt)`

Asserts that the value of the first expression is greater than the value of the second expression.

Part 3. Apply (15 min)

Start time: _____

8. Create a unit test for a Month struct that is stored inside Month.swift. Here are the expected behaviors of the structure.
- It has a property called number
 - It has a property observer for number that ensures number's value is always from 1 to 12. Otherwise, it keeps its old value.
 - It provides a computed property called name that returns the corresponding month name. For example, 1 is January, 2 is February, and so forth.
 - It has a custom initializer that does not accept any parameters, but sets the number to 1.
 - It has another custom initializer that accepts a single integer value that is assigned to the number property. If it is an invalid value (not from 1 to 12), then set the number to 1. The custom initializer omits the variable name (i.e., `var myMonth = Month(2)`)

Reflector questions

1. What was the most useful thing you learned during this session?

The most useful thing we learned was learning about XCTest.

2. What did the team do well?

We worked very well on putting together the code for the palindrome.

3. What were the challenges that the team encountered?

Coding the palindrome was difficult since we overthought the process.

4. What do you suggest the team do in the next meeting to do better?

Perhaps look at the bare minimum of what's needed before diving in and creating a super complicated process.

5. Rate your team according to the rubric below

Self-rating: 9

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0

POGIL Activity 5.1: Arrays and Dictionaries

In this activity we will learn about two data structures. Arrays allow us to store a list of information, while dictionaries allow us to map keys to values.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____ Swifters _____
 _____ 09/22/2020 _____

Date:

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Hung Cun
Presenter. Talks to the facilitator and other teams.	Scott Clary
Reflector. Considers how the team could work and learn more effectively.	Fahad Alsowaylim
Recorder. Records all answers and questions and makes the necessary submission.	T.J. Le



Part 1. Explore (20 min)

Start time: __1:45__

1. Recall at least three data structures you used to represent a collection of information in C++. For example, what data structure can store people's names, item prices, or quiz grades. You may use other programming languages that you have used more recently.

Arrays, Vectors, Hashmap, Tree, Dictionary, set, list, linked list, stack

2. Select two of these data structures and describe when they are used. Provide examples for creating it, adding, accessing, and deleting its elements if applicable. Also provide other operations on the data structure that you think is useful.

Data structure 1 name / programming language: Arrays/C++

Description: Arrays are a series of elements that are placed in specific locations which can be referenced later.

Creation/instantiation example: `int theArray[10];`

Adding elements example: `theArray[0] = 12345`

Accessing elements example: `cout << theArray[0]`

Modifying elements example: `theArray[0] += 10`

Deleting elements example (if applicable): cannot delete elements

Other operations example (if applicable):

Data structure 2 name / programming language: Vectors/C++

Description: Vectors are dynamic arrays, which means they have the same properties as arrays but they can change in size.

Creation/instantiation example: `vector<int> vector;`

Adding elements example: `vector.push_back`

Accessing elements example: `vector[12] = 22;`



Modifying elements example: `int & element = vector[12];`

Deleting elements example (if applicable): `vector.pop_back`

Other operations example (if applicable):

Part 2. Invent (20 min)

Start time: __1:53__

- Read Unit 2.5 Collections unit on ADS, specifically on arrays and dictionaries. Provide code that performs the following operations in each of these data structures.

Data structure 1 name: Array

Description: array stores an ordered list of same-typed values.

Creation/instantiation example: `var myArray: [Int] = []`

Adding elements example: `myArray.insert(1), myArray.append(2), myArray += [1,2,3,4]`

Accessing elements example: `print(myArray[4])`

Modifying elements example: `myArray[5] = 1234`

Deleting elements example (if applicable): `myArray.remove(at:2)`

Other operations example (if applicable):

Data structure 2 name: Dictionary

Description: Is a list of keys with an associated value (Like a dictionary with words to match definitions).

Creation/instantiation example: `var myDictionary: [String: Int]()`

Adding elements example: `let outdated = mydictionary.updatedValue(20, forKey: "Twenty")`

Accessing elements example: `let words = Array(myDictionary.keys)`

Modifying elements example: `let definition = Array(myDictionary.words)`



Deleting elements example (if applicable): if let oldValue = myDictionary.removeValue(forKey: "Twenty")

Other operations example (if applicable):

Part 3. Apply (15 min)

Start time: __2:00__

4. Create a structure called Phone. It has two properties, phoneNumber and contacts. phoneNumber should store the owner's number and contacts should store the names and numbers of the owner's contacts.

Create a call method that accepts a name as its parameter. The method will look for that name in the Phone's contacts and either display "Calling XXX-XXX-XXXX", where the X's are the contacts number or display "Unknown contact".

Hint: See page 177 of the ADS book.

```
struct Phone{  
  
    var phoneNumber= 0  
    //contacts should be a dictionary  
  
    var contacts: [String: Int]()  
    Func call(theName: String)  
    {  
        If let number = contacts[theName] {n  
        print("Calling \(number)"  
        }  
        Else{  
        print("Unknown contact")  
        }  
    }  
}
```

Reflector questions

1. What was the most useful thing you learned during this session?

Dictionaries

2. What did the team do well?

Explain to each other

3. What were the challenges that the team encountered?

We were able to explain to each other

4. What do you suggest the team do in the next meeting to do better?

Fahad should get a better headset

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0



POGIL Activity 5.2: Iterating through collections

In this activity we will learn about loops in Swift, but more importantly how we can use them to access information stored inside collections.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.



POGIL Template by Paul Salvador Inventado is licensed under a Creative Commons

Attribution-NonCommercial-ShareAlike 4.0 International License. Based on a work at «<http://bit.ly/2LtrlKv>».

56 of

Team name: _Swifters_____Date: __09/24/2020__

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	T.J. Le
Presenter. Talks to the facilitator and other teams.	Hung Cun
Reflector. Considers how the team could work and learn more effectively.	Scott Clary
Recorder. Records all answers and questions and makes the necessary submission.	Fahad Alsowaylim



Part 1. Explore (10 min)

Start time: 1:40____

1. Run the following programs on Swift playground, [CS50 Sandbox](#), or [repl.it](#) and provide the output.

Code	Output
<pre>// Loop 1 for index in 1...10 { if index % 2 == 0 { print(index) } }</pre>	<pre>2 4 6 8 10</pre>
<pre>// Loop 2 let expenses = [3.99, 10.58, 25.00, 15.50] for expense in expenses { print(expense) }</pre>	<pre>3.99 10.58 25.00 15.50</pre>
<pre>// Loop 3 let expenses = [3.99, 10.58, 25.00, 15.50] for (index,expense) in expenses.enumerated() { print("\(index): \(expense)") }</pre>	<pre>0: 3.99 1: 10.58 2: 25.0 3: 15.5</pre>
<pre>// Loop 4 let grades = ["A":90, "B": 80, "C": 70, "D": 60, "F" :0] for (letter, grade) in grades { print("\(letter) > \(grade)") }</pre>	<pre>A >90 C >70 B >80 D >60 F >0</pre>

Part 2. Invent (15 min)

Start time: _1:42_____

2. Describe each loop and when it should be used.

Loop 1



POGIL Template by Paul Salvador Inventado is licensed under a Creative Commons

Attribution-NonCommercial-ShareAlike 4.0 International License. Based on a work at «<http://bit.ly/2LtrlKv>».

58 of

This loop will print the even numbers in 1 through 10 (1, 2, 3, 9, 10). It could also be used to access certain indices in an array.

Loop 2

This loop will print the expenses in the array expense, it should be used when dealing with array to print out all the elements of the array.

Loop 3

This loop will print out the indexes and the expenses accordingly in the array expense. This should be used when you want print out the elements in the array and theirs indexes.

Loop 4

This loop by default prints out the letter and corresponding grade level based on the input. This should be used when you want to print out a specific variable with its corresponding data in an array.

Part 3. Apply (10 min)

Start time: 1:47__

- Update your Phonebook struct to provide a display member function that will display all its contacts. Make sure that both names and numbers are shown on screen.

```
struct Phone {
    var phoneNumber: Int
    var contacts: [String: Int]

    func displayMember() {
        for (name, number) in contacts {
            print("\(name) -- \(number)")
        }
    }

    func call(_ name: String) {
        if let number = contacts[name] {
            print("Calling \(number)")
        } else {
```



```

        print ("Unknown contact")
    }
}
}

```

Reflector questions

1. What was the most useful thing you learned during this session?

Loops

2. What did the team do well?

We did well in analyzing how each loop worked.

3. What were the challenges that the team encountered?

No challenges were encountered.

4. What do you suggest the team do in the next meeting to do better?

Nothing, we did fairly well.

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
Answered all problems in the worksheet	10



Partially answered the problems in the worksheet	8
Did not answer the worksheet	0

POGIL Activity 7.1: Optionals

In this activity we will learn about optionals. Optionals provide a way for us to elegantly deal with nil values.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____ Tiny Titans _____
_____ 10/6/20 _____

Date:



POGIL Template by Paul Salvador Inventado is licensed under a Creative Commons

Attribution-NonCommercial-ShareAlike 4.0 International License. Based on a work at «<http://bit.ly/2LtrlKv>».

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Hung Cun
Presenter. Talks to the facilitator and other teams.	Scott Clary
Reflector. Considers how the team could work and learn more effectively.	TJ Le
Recorder. Records all answers and questions and makes the necessary submission.	TJ Le / Fahad alsowaylim

Part 1. Explore (5 min)

Start time: __1:54__

1. Your group will be assigned one of the following topics: (1) Functions and optionals, (2) Failable initializers, and (3) Optional Chaining. Read the corresponding section in Section 3.1 of the ADS book and discuss as a group.

Part 2. Invent (15 min)

Start time: _____

2. What is the problem/issue/goal that your assigned topic addresses?

Allows for parameters to be nil in a function, allow return type of a function to be nil.

3. Describe the syntax for using the feature.

put "?" after the type (e.g. String?)

4. Provide an example that uses the feature.



```
Func printFullName(firstN: String, middleN: String?, lastN: String)
```

Part 3. Apply (30 min)

Start time: 2:40

5. Create the two structures described below:

a. Website

Create a Website structure that stores the title and URL of a website.

b. SocialMediaAccount

Create a SocialMediaAccount structure that stores the social media name (e.g., Facebook, Instagram, TikTok), username, password, and website.

The password should have 4 - 8 characters and the website can be optional. Design the initializer to take values for all the properties described above. However, if the password does not follow the requirements, return a nil object to indicate that the account cannot be created. Also make sure that you are able to create an account with a nil website. Hint: You can retrieve each character of a String, but that returns a Character object. You need to convert it to a String before converting it to an Int (see pg. 303 of the ADS).

Create a WebSite and SocialMediaAccount instance. Provide any values you prefer. If the password is invalid, display "Invalid Password." Display the social media name, username, and website link (URL) of the account. Use optional chaining to display the website link to simplify your code.

Use Swift Playground to implement your solution. Run the program and take a screenshot of the code and output. Place the screenshot below.

```
Struct Website {
    var title: String
    var url: String
}

Struct SocialMediaAccount {
    var socialMediaName: String
    var username: String
    var password: String
    var website: Website?

    Init?(socialMediaName: String, username: String, password: String, website:
```



```
Website?) {
    if password.count < 4 || password.count > 8 {
        return nil
    } else {
        self.socialMediaName = socialMediaName
        self.username = username
        self.password = password
        self.website = website
    }
}

let web = Website(title: "Pogil Group 3", url:
"https://docs.google.com/document/d/1TxtcUAE1i41-E5keSZzwBU6mo4dqT0JZ-yRY3ilCyXk/edit?ts=5f612bfe#")

let possibleaccount = SocialMediaAccount(socialMediaName: "Google", username:
"cspc223w", password: "cspc223w", website: web)
if let account = possibleaccount {
    print("socialMediaName: \(account.socialMediaName), username:
\(account.username)")
    if let site = account.website?.url {
        print(site)
    }
} else {
    print("Invalid password")
}
```

Reflector questions

1. What was the most useful thing you learned during this session?

The most important thing we learned in this lesson was about functions and optionals.

2. What did the team do well?

We did well in discussing the definition and meaning of what the optionals do.



3. What were the challenges that the team encountered?

No challenges

4. What do you suggest the team do in the next meeting to do better?

I think we did fairly well

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0

POGIL Activity 8.1: Guard

In this activity we will learn about guards in Swift, which are special constructs that promote early exit. Using early exit is a best practice in software development that promotes readability.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____

Date: ____10/13/20____

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Fahad
Presenter. Talks to the facilitator and other teams.	Ismail
Reflector. Considers how the team could work and learn more effectively.	Mikias
Recorder. Records all answers and questions and makes the necessary submission.	Kevin



Part 1. Explore (10 min)

Start time: ____1:45____

1. Explain the similarities and differences between the two functions below.

Code 1

```
func isValid(password: String) -> Bool {
    if password.count > 0 && password.count < 9 {
        var hasNumber = false
        for char in password {
            if char.isNumber {
                hasNumber = true
            }
        }
        if hasNumber {
            var hasLetter = false
            for char in password {
                if char.isLetter {
                    hasLetter = true
                }
            }
            if hasLetter {
                return true
            } else {
                return false
            }
        } else {
            return false
        }
    } else {
        return false
    }
}
```

Code 2

```
func isValid(password: String) -> Bool {
    if password.count < 1 || password.count > 8 {
        return false
    }
    var hasNumber = false
    for char in password {
        if char.isNumber {
            hasNumber = true
        }
    }
    if !hasNumber {
```

```
    return false;
}
var hasLetter = false
for char in password {
    if char.isLetter {
        hasLetter = true
    }
}
if !hasLetter {
    return false;
}

return true;
}
```

Similarities

Both takes a boolean value and returns a string
For loop is the same

Differences

Code 1 is checking length of password inside of password (count 1-8)
Code 2 is checking if the password length is less than one and greater than 8
If hasletter is inverted

Part 2. Invent (5 min)

Start time: ____1:55____

2. What are the advantages of using Code 2?

Code 2 is **significantly** easier to read. If else statement in Code 1 is confusing.



Part 3. Apply (20 min)

Start time: ____2:20____

3. Modify the code in #1 to use the guard statement

```
func isValid(password: String) -> Bool {
    guard password.count > 0 && password.count < 9 else {
        print("too long or short")
        return false
    }

    var hasNumber = false
    for char in password {
        if char.isNumber {
            hasNumber = true
        }
    }

    guard hasNumber else {
        print("no number!")
        return false
    }

    var hasLetter = false
    for char in password {
        if char.isLetter {
            hasLetter = true
        }
    }

    guard hasLetter else {
        print("no letter!")
        return false
    }

    return true
}
```

Reflector questions

1. What was the most useful thing you learned during this session?

Guard statements are the most useful thing we learned today.



2. What did the team do well?

The team (ismail) did well communicating any problems we had.

3. What were the challenges that the team encountered?

Initially, rewriting code 1 as a guard statement was a little confusing.

4. What do you suggest the team do in the next meeting to do better?

We did pretty well today overall, nothing to really improve on.

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	10
Did not answer the worksheet	0

POGIL Activity 8.2: Enumeration

In this activity we will learn to use Enumerations to make code readable. Specifically it allows us to define variables that can take a finite set of values.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____ Date: _____

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	
Presenter. Talks to the facilitator and other teams.	
Reflector. Considers how the team could work and learn more effectively.	
Recorder. Records all answers and questions and makes the necessary submission.	



Part 1. Explore (10 min)

Start time: _____

Observe the code below while focusing on the role property of the User struct.

```
struct Post {
    var title: String
    var author: String
    var content: String
}

struct User {
    var username: String
    var role: String
    var feed : [Post]

    init(username: String, role: String) {
        self.username = username
        self.role = role
        self.feed = []
    }

    mutating func add(post: Post) {
        feed.append(post)
    }

    func displayPosts() {
        for post in feed {
            if role == "Admin" {
                print(post.title)
                print(post.author)
                print(post.content)
            } else if role == "Member" {
                print(post.title)
                print(post.content)
            } else if role == "Guest" {
                print(post.title)
            }
            print("\n")
        }
    }
}

var post1 = Post(title: "Faculty and Staff Receive Nearly $9
Million in Grants, Contracts in Q1", author:
"strategiccomm@fullerton.edu", content: "Faculty and staff
garnered close to $9 million in funding during the first quarter
```



```
of the university's 2020-21 fiscal year.")

var post2 = Post(title: "Students Fired Up to Develop Miniature
Satellite to Fight Wildfires", author:
"strategiccomm@fullerton.edu", content: "As wildfires continue
to cause destruction across California, a team of engineering
students is exploring a solution to fight the devastating
infernos by designing and building a miniaturized satellite
known as a CubeSat.")

var post3 = Post(title: "Faculty Showcase Expertise Through
Publications, Presentations and Awards", author:
"strategiccomm@fullerton.edu", content: "See a listing of the
recent publications, presentations and awards from Cal State
Fullerton faculty.")

var me = User(username: "Paul", role: "Admin")
me.add(post: post1)
me.add(post: post2)
me.add(post: post3)

me.displayPosts()
```

1. What are the potential issues with the current code design?

Current code design looks like a short essay. Very bad readability.

Part 2. Invent (5 min)

Start time: _____

2. Read unit 3.5 Enumeration in the ADS book. How can Enumeration potentially address the issues you identified in #1?



Part 3. Apply (20 min)

Start time: _____

3. Create an Enumeration and modify the code in #1 to use your Enumeration.

Reflector questions

1. What was the most useful thing you learned during this session?

2. What did the team do well?

3. What were the challenges that the team encountered?

4. What do you suggest the team do in the next meeting to do better?

5. Rate your team according to the rubric below

Self-rating:

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0



POGIL Activity 9.1: Protocols

In this activity we will learn about protocols that allow us to require functionalities from classes. Protocols allow us to design code that is easier to adapt and scale.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____ Date: _____


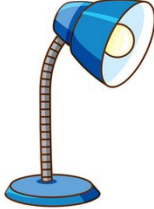






Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	
Presenter. Talks to the facilitator and other teams.	
Reflector. Considers how the team could work and learn more effectively.	
Recorder. Records all answers and questions and makes the necessary submission.	



Part 1. Explore (5 min)

Start time: __1:44__

Observe the images below

Port	Devices
	  
	  

1. Why is it possible to connect different devices to the same port?

Yup. The devices that can be plugged into the same port have the same type of plug, because that plug is seen as a “universal standard”

Part 2. Invent (10 min)

Start time: __1:45__

2. Let's imagine that we are developing a new standard so that we can design a robot that uses different cooking appliances to cook food. What do you think are the common properties and functionalities that cooking appliances share? (e.g., microwave, oven, toaster, air fryer).

True. Timer, cook, modes, start/end

Part 3. Apply (25 min)

Start time: __2:17__

3. Design a `CookingAppliance` protocol and define properties and functions according to what you identified in #2. Make sure the protocol contains at least one function named `start`.

```
protocol CookingAppliance {
    var timer: Int { get set }

    func start()
}
```

4. Select one of the cooking appliances you considered in #2 and create a struct to represent it. Make sure it uses the `CookingAppliance` and `CustomStringConvertible` protocols. You can print values on the screen to simulate the cooking process.

```
struct Cooker: CookingAppliance, CustomStringConvertible {
    var timer: Int
    var description: String

    mutating func start() {
        while timer > 0 {
            self.timer -= 1
            description = "Time remaining: \(self.timer)"
            print(self)
        }
    }
}
```



```

        print("Done!")
    }

    init() {
        timer = 0
        description = ""
    }
}

```

5. Will your code work with the following struct? Explain why.

```

struct RoboChef {
    var name: String
    var appliance: CookingAppliance
    mutating func cook(food: String) {
        print("Cooking \(food)");
        appliance.timer = 15;
        appliance.start()
    }
}

// Call the constructor of your CookingAppliance beside the =
// to create an instance of your struct

let myAppliance =
var chef = RoboChef(name: "Gordon", appliance: myAppliance)
chef.cook(food: "potato")

```

Explanation

Yes, our code would work with this RoboChef struct, simply because the struct and appliance that the code has conforms with the CookingAppliance and Cooker struct that we designed.

6. If anyone now or in the future decides to create a struct that uses the CookingAppliance protocol, will that new class also work with the RoboChef struct in #5? Why?

It will work, because the CookingAppliance is basically a new standard that we put into place, and since RoboChef works with CookingAppliance, anything else that uses the CookingAppliance protocol will work with RoboChef as well.

Reflector questions

1. What was the most useful thing you learned during this session?

We learned about protocols

2. What did the team do well?

Solving/explaining together

3. What were the challenges that the team encountered?

2 member had broken mics

4. What do you suggest the team do in the next meeting to do better?

Get new headsets lol

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0



POGIL Activity 9.2: Saving data

In this activity we will learn how to encode, decode, store, and load data. You are expected to use your prior understanding of Swift programming constructs and practice understanding documentation.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____ Date: _____

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	
Presenter. Talks to the facilitator and other teams.	
Reflector. Considers how the team could work and learn more effectively.	
Recorder. Records all answers and questions and makes the necessary submission.	



Part 1. Explore (15 min)**Start time:** _____

Read pp. 659 - 666 in the App Development with Swift Book. Specifically, read from Encoding and Decoding with Codable to Saving an Array of Model Data. You will need to recall and apply some of the concepts we have discussed in class to understand some of the discussions.

Part 2. Invent (15 min)**Start time:** _____

1. Provide code that would use a `PropertyListEncoder` to encode data.

```
let propertyListEncoder = PropertyListEncoder()
if let encodedNote = try? propertyListEncoder.encode(newNote) {
    print(encodedNote)
}
```

2. Provide code that would use a `PropertyListDecoder` to decode an encoded object.

```
let propertyListDecoder = PropertyListDecoder()
if let decodedNote = try? propertyListDecoder
    .decode(Note.self, from: encodedNote) {
    print(decodedNote)
}
```

3. What is a sandbox and how is it relevant to files in Swift?

A sandbox is an environment that each app gets its own of, and it gets to create, modify, or delete data. It's relevant to Swift because iOS may allow access to resources OUTSIDE the sandbox, but ONLY when your app receives explicit permission from the user to do so.

4. Provide code that would represent a file URL in the user's document directory (any file name will do).

```
let documents = FileManager.default.urls(for:
.documentDirectory, in: .userDomainMask).first!
```

5. Provide code that would write data into a file in the user's document directory (any file name will do).

```
struct Note: Codable {
    let title: String
    let text: String
    let timestamp: Date
}

let newNote = Note(title: "Grocery run", text: "Pick up
mayonnaise, mustard, lettuce, tomato, and pickles.", timestamp:
Date())

let documentsDirectory =
FileManager.default.urls(for: .documentDirectory,
in: .userDomainMask).first!
let archiveURL =
documentsDirectory.appendingPathComponent("notes_test").
appendingPathExtension("plist")

let propertyListEncoder = PropertyListEncoder()
let encodedNote = try? propertyListEncoder.encode(newNote)
```

6. Provide code that would read data from a file in the user's document directory into a variable.

```
let propertyListDecoder = PropertyListDecoder()
if let retrievedNoteData = try? Data(contentsOf: archiveURL),
    let decodedNote = try?
propertyListDecoder.decode(Note.self,
from: retrievedNoteData) {
    print(decodedNote)
}
```

Part 3. Apply (15 min)

Start time: _____

Consider the Contact struct below to answer the following questions.

```
struct Contact {  
    var name: String  
    var number: String  
}
```

7. Write code that would store a Contact instance called myContact into a file called contacts.plist.

```
let myContact = Contact(name: "Ismail", number: "888")  
let documentsDirectory = FileManager.default.urls(for:  
    .documentDirectory, in: .userDomainMask).first!  
let archiveURL =  
    documentsDirectory.appendingPathComponent("contacts").appendingPathExtension("plist")  
  
let propertyListEncoder = PropertyListEncoder()  
let encodedNote = try? propertyListEncoder.encode(myContact)
```

8. Write code that would read a Contact object from contacts.plist and store it in a Contact instance called loadedContact.

```
let propertyListDecoder = PropertyListDecoder()  
if let retrievedContactData = try? Data(contentsOf:  
    archiveURL),  
    let decodedNote = try?  
        propertyListDecoder.decode(Contact.self,  
            from: retrievedContactData) {  
    print(decodedNote)  
}
```

Reflector questions

1. What was the most useful thing you learned during this session?

Learning how to save data

2. What did the team do well?

Going and reading the chapter while communicating our thoughts

3. What were the challenges that the team encountered?

nah

4. What do you suggest the team do in the next meeting to do better?

For everyone to be here >:(

5. Rate your team according to the rubric below

Self-rating: 9.95

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0

POGIL Activity 10.1: Closures

In this activity we will learn about closures. Closures allow us to define functions that we can store in variables together with other definitions that are available within its scope. Closures enable commonly used designs such as function callbacks, retrieving appropriate implementations, and the strategy design pattern.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____

Date: _____

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	
Presenter. Talks to the facilitator and other teams.	
Reflector. Considers how the team could work and learn more effectively.	
Recorder. Records all answers and questions and makes the necessary submission.	



Part 1. Explore (5 min)

Start time: _____

You are assigned to read and understand one of the following functions that use closures from the ADS book: map (pp. 789 - 791), filter (pp. 791 - 793), reduce (pp. 793 - 794). Make sure to understand the concept as you will be asked to explain it in class.

Part 2. Invent (5 min)Start time: 7:28_____

1. Provide sample code for using the function you were assigned in part 1.

```
let numbers = [4, 8, 15, 16, 23, 42]
let numbersLessThan20 = numbers.filter { (number) -> Bool in
    return number < 20
}

print (numbersLessThan20)
```

Part 3. Apply (30 min)Start time: 7:30_____

2. Let's assume that your grocery is having a sale and all products that are \$0.50 and below are free! A Product struct and driver code is provided for you below. You must use the map, filter, and reduce functions to write a computeSalePrice function. The function accepts an array of Products and returns a Double value. It should perform the following steps:
 - a. Extract the prices of each element in the list of products
 - b. Select only prices that are \$0.50
 - c. Add all prices
 - d. Return the total price

```
struct Product {
    var name: String
    var price: Double
}
```



```

var quantity: Int
var totalPrice: Double {
    Double(quantity) * price
}
}

func computeSalePrice(products: [Product]) -> Double {
    let finalProducts = products.filter { (product) -> Bool in
        return product.price > 0.50
    }

    return finalProducts.reduce(0) {$0 + $1.price}
}

var milk = Product(name: "Milk", price: 3.99, quantity: 2)
var gum = Product(name: "Gum", price: 0.50, quantity: 1)
var roastedChicken = Product(name: "Roasted Chicken",
                             price: 7.99, quantity: 2)

var groceries = [milk, gum, roastedChicken]
let finalPrice = computeSalePrice(products: groceries)
print(finalPrice)

```

3. Create a `displayProducts` function that will display the contents of an array of `Products`. We will use closures to let developers control how to display the values on the screen (or some other form of output). The `displayProducts` function should accept two parameters, the array of `Products` with an omitted argument and a closure. The closure should accept three parameters, a `String`, a `Double`, and an `Int`. It should not return anything. The `String` parameter refers to the name of the product, the `Double` is the price and the `Int` is the quantity. The example below shows how your function might be used.

```

func displayProduct

// Sample usage of the displayProducts function

var products: [Product] = []
products.append(Product(name: "Milk", price: 3.99,
                        quantity: 1))
products.append(Product(name: "Gum", price: 0.50, quantity: 1))

```



```
displayProducts(products: products) {
    (name, price, quantity) -> Void in
    print("\(quantity) x \(name) : $\(price)")
}
```

Reflector questions

1. What was the most useful thing you learned during this session?

You can pass functions like variables

2. What did the team do well?

Got it done

3. What were the challenges that the team encountered?

none

4. What do you suggest the team do in the next meeting to do better?

none

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
----------	-------



Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0



POGIL Activity 11.1: Extensions

In this activity we will learn about extensions. Extensions allow us to add functionalities to existing classes, structures, enumerations, or protocols. They are also use to organize code and promote readability.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____

Date: _____

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	
Presenter. Talks to the facilitator and other teams.	
Reflector. Considers how the team could work and learn more effectively.	Fahad
Recorder. Records all answers and questions and makes the necessary submission.	



Part 1. Explore (10 min)

Start time: __1:40__

Read Lesson 5.2: Extensions of the App Development with Swift book (pp. 796 - 801). Pay particular attention to the purpose of extensions and their syntax.

Part 2. Invent (10 min)

Start time: __1:43__

1. What are the two things extensions are mostly used for? Describe each one.

Purpose 1

Adding functionality to types you can't edit directly, such as types defined in the Swift standard library

Purpose 2

Organizing code into logical chunks

Part 3. Apply (15 min)

Start time: _____

2. Extend the String structure to provide a titlecased() function. This modifies the String so that the first letter of each word is converted to uppercase. Let's assume that each word is separated by a space. For example, "hello world" becomes "Hello World".

```
extension String {  
    mutating func titlecased() -> String {  
        return self.capitalized  
    }  
}
```

3. Refactor the SoccerGame struct by moving its methods into an extension.



```

struct SoccerGame {
    var homeTeam: String
    var visitingTeam: String
    var schedule: String

    var duration: Double
    var homeTeamScore: Int
    var visitingTeamScore: Int

    init(homeTeam: String, visitingTeam: String,
         schedule: String) {
        self.homeTeam = homeTeam
        self.visitingTeam = visitingTeam
        self.schedule = schedule
        homeTeamScore = 0
        visitingTeamScore = 0
        duration = 90
    }

    func showGameInfo() -> Void {
        print("\(homeTeam) vs. \(visitingTeam)")
        print("\(schedule)")
    }

    func showScore() -> Void {
        print("\(homeTeam) \(homeTeamScore) - \(visitingTeamScore)
              \(visitingTeam)")
        print("\(duration)")
    }

    func showWinner() -> Void {
        guard duration == 0 else {
            print("Match between \(homeTeam) and \(visitingTeam) has
                  not yet started.")
            return
        }
        if homeTeamScore > visitingTeamScore {
            print("\(homeTeam) wins over \(visitingTeam)")
            print("\(homeTeamScore) - \(visitingTeamScore)")
        } else {
            print("\(visitingTeam) wins over \(homeTeam)")
            print("\(visitingTeamScore) - \(homeTeamScore)")
        }
    }
}

```

Refactored version using extensions

```

struct SoccerGame {
    var homeTeam: String

```



```

var visitingTeam: String
var schedule: String

var duration: Double
var homeTeamScore: Int
var visitingTeamScore: Int

init(homeTeam: String, visitingTeam: String,
      schedule: String) {
    self.homeTeam = homeTeam
    self.visitingTeam = visitingTeam
    self.schedule = schedule
    homeTeamScore = 0
    visitingTeamScore = 0
    duration = 90
}

extension SoccerGame {
    func showGameInfo() -> Void {
        print("\(homeTeam) vs. \(visitingTeam)")
        print("\(schedule)")
    }

    func showScore() -> Void {
        print("\(homeTeam) \(homeTeamScore) -
\(visitingTeamScore) \(visitingTeam)")
        print("\(duration)")
    }

    func showWinner() -> Void {
        guard duration == 0 else {
            print("Match between \(homeTeam) and \(visitingTeam)
has not yet started.")
            return
        }
        if homeTeamScore > visitingTeamScore {
            print("\(homeTeam) wins over \(visitingTeam)")
            print("\(homeTeamScore) - \(visitingTeamScore)")
        } else {
            print("\(visitingTeam) wins over \(homeTeam)")
            print("\(visitingTeamScore) - \(homeTeamScore)")
        }
    }
}

```

Reflector questions

1. What was the most useful thing you learned during this session?

Use extensions to refactor

2. What did the team do well?

Explained to each other

3. What were the challenges that the team encountered?

none

4. What do you suggest the team do in the next meeting to do better?

Same thing

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0

POGIL Activity 11.1: Extensions

In this activity we will learn about extensions. Extensions allow us to add functionalities to existing classes, structures, enumerations, or protocols. They are also use to organize code and promote readability.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____

Date: _____

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	
Presenter. Talks to the facilitator and other teams.	
Reflector. Considers how the team could work and learn more effectively.	
Recorder. Records all answers and questions and makes the necessary submission.	



Part 1. Explore (10 min)

Start time: _____

Read Lesson 5.2: Extensions of the App Development with Swift book (pp. 796 - 801). Pay particular attention to the purpose of extensions and their syntax.

Part 2. Invent (10 min)

Start time: _____

1. What are the two things extensions are mostly used for? Describe each one.

Purpose 1
Purpose 2

Part 3. Apply (15 min)

Start time: _____

2. Extend the String structure to provide a titlecased() function. This modifies the String so that the first letter of each word is converted to uppercase. Let's assume that each word is separated by a space. For example, "hello world" becomes "Hello World".

--

3. Refactor the SoccerGame struct by moving its methods into an extension.

<pre>struct SoccerGame { var homeTeam: String var visitingTeam: String</pre>
--

```
var schedule: String

var duration: Double
var homeTeamScore: Int
var visitingTeamScore: Int

init(homeTeam: String, visitingTeam: String,
      schedule: String) {
    self.homeTeam = homeTeam
    self.visitingTeam = visitingTeam
    self.schedule = schedule
    homeTeamScore = 0
    visitingTeamScore = 0
    duration = 90
}

func showGameInfo() -> Void {
    print("\(homeTeam) vs. \(visitingTeam)")
    print("\(schedule)")
}

func showScore() -> Void {
    print("\(homeTeam) \(homeTeamScore) - \(visitingTeamScore)
          \(visitingTeam)")
    print("\(duration)")
}

func showWinner() -> Void {
    guard duration == 0 else {
        print("Match between \(homeTeam) and \(visitingTeam) has
              not yet started.")
        return
    }
    if homeTeamScore > visitingTeamScore {
        print("\(homeTeam) wins over \(visitingTeam)")
        print("\(homeTeamScore) - \(visitingTeamScore)")
    } else {
        print("\(visitingTeam) wins over \(homeTeam)")
        print("\(visitingTeamScore) - \(homeTeamScore)")
    }
}
}
```

Refactored version using extensions



Reflector questions

1. What was the most useful thing you learned during this session?

2. What did the team do well?

3. What were the challenges that the team encountered?

4. What do you suggest the team do in the next meeting to do better?

5. Rate your team according to the rubric below

Self-rating:

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0

POGIL Activity 12.1: Working with the Web

In this activity we will learn to send and receive information through the internet. This involves a complex process that we are now able to explain using the different concepts we have learned such as optionals, enumeration, guard, protocols, and closures.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____ Date: _____

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	
Presenter. Talks to the facilitator and other teams.	
Reflector. Considers how the team could work and learn more effectively.	Fahad
Recorder. Records all answers and questions and makes the necessary submission.	



Part 1 & 2. Explore and Invent (80 min) Start time: 2:00

1. Copy the code below into Swift Playground and run the program. Describe the general purpose of the code and its output. Explain how the code works and focus on the concepts we discussed in class (e.g., optionals, enumeration, guard, protocols, and closures). Feel free to read pp. 843 - 849 from the ADS for reference.

```
import Foundation
import PlaygroundSupport

PlaygroundPage.current.needsIndefiniteExecution = true

let url = URL(string:
"https://apps.apple.com/us/app/keynote/id409183694?mt=12")!
let task = URLSession.shared.dataTask(with: url) {
    (data, response, error) in
    if let data = data,
        let string = String(data: data, encoding: .utf8) {
        print(string)
    }
    PlaygroundPage.current.finishExecution()
}

task.resume()
```

Code description and output

The output for this code seems to be the html code for the URL's page. (not sure if this is correct tho)

Code explanation

let url = Url(string:) This part of the code creates a URL object that points to the website in question. ! ← at the end of the let url line “force-unwraps” the optional URL.

let task = URLSession... This line creates a the URLSession class and then uses the “shared”

- Copy the code below into Swift Playground and run the program. Describe the general purpose of the code and its output. Explain how the code works and focus on the concepts we discussed in class (e.g., optionals, enumeration, guard, protocols, and closures). You do not need to explain code that has already been discussed in previous questions. The changes in the code are highlighted in yellow. Feel free to read pp. 849 - 853 from the ADS for reference.

```
import Foundation
import PlaygroundSupport

PlaygroundPage.current.needsIndefiniteExecution = true

extension URL {
    func withQueries(_ queries: [String: String]) -> URL? {
        var components = URLComponents(url: self,
            resolvingAgainstBaseURL: true)
        components?.queryItems = queries.map {
            URLQueryItem(name: $0.0, value: $0.1)
        }
        return components?.url
    }
}

let baseURL = URL(string:
    "https://api.nasa.gov/planetary/apod")!

let query: [String: String] = [
    "api_key": "DEMO_KEY",
    "date": "2020-11-09"
]

let url = baseURL.withQueries(query)!
let task = URLSession.shared.dataTask(with: url) {
    (data, response, error) in
    if let data = data,
        let string = String(data: data, encoding: .utf8) {
        print(string)
    }
    PlaygroundPage.current.finishExecution()
}

task.resume()
```

Code description and output

Displays a quote that is copyrighted from a person named max rive and the date of



when it was posted? As well as the link that redirects the user to the images

Code explanation

This code uses Queries that allows the code to fetch data that is provided from the url, the api key that is necessary to access the website

Extension url = add onto the url by using a user created extension to include the queries

- Copy the code below into Swift Playground and run the program. Describe the general purpose of the code and its output. Explain how the code works and focus on the concepts we discussed in class (e.g., optionals, enumeration, guard, protocols, and closures). You do not need to explain code that has already been discussed in previous questions. The changes in the code are highlighted in yellow. Feel free to read pp. 859 - 862 from the ADS for reference.

```
import Foundation
import PlaygroundSupport

PlaygroundPage.current.needsIndefiniteExecution = true

extension URL {
    func withQueries(_ queries: [String: String]) -> URL? {
        var components = URLComponents(url: self,
            resolvingAgainstBaseURL: true)
        components?.queryItems = queries.map {
            URLQueryItem(name: $0.0, value: $0.1)
        }
        return components?.url
    }
}

let baseURL = URL(string:
    "https://api.nasa.gov/planetary/apod")!

let query: [String: String] = [
    "api_key": "DEMO_KEY",
    "date": "2020-11-09"
]

let url = baseURL.withQueries(query)!
```

```

let task = URLSession.shared.dataTask(with: url) {
    (data, response, error) in
    let jsonDecoder = JSONDecoder()
    if let data = data,
        let photoDictionary = try? jsonDecoder.decode([String:
            String].self, from: data) {
        print(photoDictionary)
        print()
        print(photoDictionary["title"])
        print(photoDictionary["explanation"])
        print(photoDictionary["url"])
    }
    PlaygroundPage.current.finishExecution()
}

task.resume()

```

Code description and output

The new output from this code displays the title of the article as well as the explanation of the image it received from the website.

Code explanation

Contains a closure that decodes a json file and prints the title, explanation, and url from the link provided

4. Copy the code below into Swift Playground and run the program. Describe the general purpose of the code and its output. Explain how the code works and focus on the concepts we discussed in class (e.g., optionals, enumeration, guard, protocols, and closures). You do not need to explain code that has already been discussed in previous questions. The changes in the code are highlighted in yellow. Feel free to read pp. 862 - 868 from the ADS for reference.

```

import Foundation
import PlaygroundSupport

PlaygroundPage.current.needsIndefiniteExecution = true

extension URL {
    func withQueries(_ queries: [String: String]) -> URL? {
        var components = URLComponents(url: self,

```



```

        resolvingAgainstBaseURL: true)
        components?.queryItems = queries.map {
URLQueryItem(name: $0.0, value: $0.1) }
        return components?.url
    }
}

struct PhotoInfo: Codable {
    var title: String
    var description: String
    var url: URL
    var copyright: String?

    enum CodingKeys: String, CodingKey {
        case title
        case description = "explanation"
        case url
        case copyright
    }

    init(from decoder: Decoder) throws {
        let valueContainer = try decoder.container(keyedBy:
            CodingKeys.self)
        self.title = try valueContainer.decode(String.self,
            forKey: CodingKeys.title)
        self.description = try
            valueContainer.decode(String.self, forKey:
            CodingKeys.description)
        self.url = try valueContainer.decode(URL.self, forKey:
            CodingKeys.url)
        self.copyright = try?
            valueContainer.decode(String.self,
            forKey: CodingKeys.copyright)
    }
}

let baseURL = URL(string:
    "https://api.nasa.gov/planetary/apod")!

let query: [String: String] = [
    "api_key": "DEMO_KEY",
    "date": "2020-11-09"
]

let url = baseURL.withQueries(query)!
let task = URLSession.shared.dataTask(with: url) {
    (data, response, error) in
    let jsonDecoder = JSONDecoder()
    if let data = data,

```

```

        let photoInfo = try? jsonDecoder.decode(PhotoInfo.self,
            from: data) {
            print(photoInfo)
            print()
            print(photoInfo.title)
            print(photoInfo.description)
            print(photoInfo.url)
        }
        PlaygroundPage.current.finishExecution()
    }

    task.resume()

```

Code description and output

This code outputs all of the string on the URL's page. There is an image with a title and description on the webpage. Although for the output, the terminal can only output the description of the image but not the image itself.

Code explanation

5. Copy the code below into Swift Playground and run the program. Describe the general purpose of the code and its output. Explain how the code works and focus on the concepts we discussed in class (e.g., optionals, enumeration, guard, protocols, and closures). You do not need to explain code that has already been discussed in previous questions. The changes in the code are highlighted in yellow. Feel free to read pp. 869 - 875 from the ADS for reference.

```

import Foundation
import PlaygroundSupport

PlaygroundPage.current.needsIndefiniteExecution = true

extension URL {
    func withQueries(_ queries: [String: String]) -> URL? {
        var components = URLComponents(url: self,
            resolvingAgainstBaseURL: true)
        components?.queryItems = queries.map {
            URLQueryItem(name: $0.0, value: $0.1)
        }
    }
}

```

```

        return components?.url
    }
}

struct PhotoInfo: Codable {
    var title: String
    var description: String
    var url: URL
    var copyright: String?

    enum CodingKeys: String, CodingKey {
        case title
        case description = "explanation"
        case url
        case copyright
    }

    init(from decoder: Decoder) throws {
        let valueContainer = try decoder.container(keyedBy:
            CodingKeys.self)
        self.title = try valueContainer.decode(String.self,
            forKey: CodingKeys.title)
        self.description = try
            valueContainer.decode(String.self, forKey:
            CodingKeys.description)
        self.url = try valueContainer.decode(URL.self, forKey:
            CodingKeys.url)
        self.copyright = try?
            valueContainer.decode(String.self,
            forKey: CodingKeys.copyright)
    }
}

func fetchPhotoInfo(completion: @escaping (PhotoInfo?) -> Void)
{
    let baseURL = URL(string:
        "https://api.nasa.gov/planetary/apod")!

    let query: [String: String] = [
        "api_key": "DEMO_KEY",
        "date": "2020-11-09"
    ]

    let url = baseURL.withQueries(query)!
    let task = URLSession.shared.dataTask(with: url) { (data,
        response, error) in
        let jsonDecoder = JSONDecoder()
        if let data = data,
            let photoInfo = try?

```

```
        jsonDecoder.decode(PhotoInfo.self, from: data) {
            completion(photoInfo)
        } else {
            print("Either no data was returned, or data was not
properly decoded.")
            completion(nil)
        }
        PlaygroundPage.current.finishExecution()
    }

    task.resume()
}

fetchPhotoInfo {
    (photoInfo) -> Void in
    if let pInfo = photoInfo {
        print(pInfo.title)
        print(pInfo.description)
        print(pInfo.url)
    }
}
```

Code description and output

From the code, it looks like everything regarding the base url, queries, and actual decoding of JSON info grabbed was put into a function. Therefore, the data cannot be called from anywhere else in the code, other than the declaration of photoInfo.

Code explanation

The function is called, along with an if let statement making sure that the data grabbed isn't nil. Once valid data is received and decoded, it prints the title, description, and url of said info.

6. Create a simple user interface that will contain information that is retrieved from NASA's API. Specifically, create a UIImage and two labels to hold the photo of the day, its description, and copyright information. Page 885 of the ADS book provides a screenshot of a possible layout. Create and copy the code into the corresponding files specified below. For ViewController.swift, you will need to create outlets that connect the labels to your view controller. Kindly name them descriptionLabel and copyrightLabel so they work directly with the code provided.

Describe the general purpose of the code in the viewDidLoad function of ViewController.swift and its expected output. Explain how the code works and focus on the concepts we discussed in class (e.g., optionals, enumeration, guard, protocols, and closures). You do not need to explain code that has already been discussed in previous questions. The changes in the code are highlighted in yellow. Feel free to read pp. 885 - 892 from the ADS for reference.

PhotoInfo.swift

```
import Foundation

struct PhotoInfo : Codable {
    var title: String
    var description: String
    var url: URL
    var copyright: String?

    enum CodingKeys: String, CodingKey {
        case title
        case description = "explanation"
        case url
        case copyright
    }

    init(from decoder: Decoder) throws {
        let valueContainer = try decoder.container(keyedBy:
            CodingKeys.self)
        self.title = try valueContainer.decode(String.self,
            forKey: CodingKeys.title)
```



```

        self.description = try
            valueContainer.decode(String.self, forKey:
                CodingKeys.description)
        self.url = try valueContainer.decode(URL.self, forKey:
            CodingKeys.url)
        self.copyright = try?
            valueContainer.decode(String.self,
                forKey: CodingKeys.copyright)
    }
}

```

URL+Helpers.swift

```

import Foundation

extension URL {
    func withQueries(_ queries: [String: String]) -> URL? {
        var components = URLComponents(url: self,
            resolvingAgainstBaseURL: true)
        components?.queryItems = queries.map {
            URLQueryItem(name: $0.0, value: $0.1)
        }
        return components?.url
    }
}

```

PhotoInfoController.swift

```

import Foundation

class PhotoInfoController {
    func fetchPhotoInfo(completion: @escaping (PhotoInfo?) -> Void) {
        let baseURL = URL(string:
            "https://api.nasa.gov/planetary/apod")!

        let query: [String: String] = [
            "api_key": "DEMO_KEY",
            "date": "2020-11-07"
        ]

        let url = baseURL.withQueries(query)!
        let task = URLSession.shared.dataTask(with: url) {
            (data,
            response, error) in
                let jsonDecoder = JSONDecoder()
                if let data = data,
                    let photoInfo = try?
                        jsonDecoder.decode(PhotoInfo.self, from: data)
                {

```

```

        completion(photoInfo)
        print(photoInfo)
    } else {
        print("Either no data was returned, or data was
not properly decoded.")
        completion(nil)
    }
}
task.resume()
}
}

```

ViewController.swift

Don't forget to create outlets for your two labels and name them descriptionLabel and copyrightLabel. Replace the viewDidLoad function that was automatically generated with the code below.

```

let photoInfoController = PhotoInfoController()

override func viewDidLoad() {
    super.viewDidLoad()

    descriptionLabel.text = ""
    copyrightLabel.text = ""

    photoInfoController.fetchPhotoInfo { (photoInfo) in
        guard let photoInfo = photoInfo else { return }
        DispatchQueue.main.async {
            self.title = photoInfo.title
            self.descriptionLabel.text =
photoInfo.description
            if let copyright = photoInfo.copyright {
                self.copyrightLabel.text =
"Copyright \(copyright)"
            } else {
                self.copyrightLabel.isHidden = true
            }
        }
    }
}
}

```

Code description and output

The ViewController class essentially allows for the web grab code to be displayed on a simple UI. The output is what we see.

Code explanation

First, the code creates an instance of the PhotoInfoController class so that we can actually use a URL to grab some JSON decoded data. Next, using UI connections in

the storyboard, it sets two labels to have empty text, in order to prepare them for the text decoded text. The code then uses the PhotoInfoController class instance to grab data from the NASA site, and once it does, it displays that info in descriptionLabel.text, and copyrightLabel.text.

7. Replace the viewDidLoad function and add the updateUI function provided below into ViewController.swift. Describe the general purpose of the code and its output. Explain how the code works and focus on the concepts we discussed in class (e.g., optionals, enumeration, guard, protocols, and closures). You do not need to explain code that has already been discussed in previous questions. The changes in the code are highlighted in yellow. Feel free to read pp. 892 - 894 from the ADS for reference.

```

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.
    descriptionLabel.text = ""
    copyrightLabel.text = ""
    photoInfoController.fetchPhotoInfo { (photoInfo) in
        guard let photoInfo = photoInfo else { return }
        self.updateUI(with: photoInfo)
    }
}

func updateUI(with photoInfo: PhotoInfo) {
    let task = URLSession.shared.dataTask(with:
photoInfo.url) { (data,
    response, error) in
        guard let data = data,
            let image = UIImage(data: data) else { return }
        print(image)
        DispatchQueue.main.async {
            self.title = photoInfo.title
            self.descriptionLabel.text =
photoInfo.description
            if let copyright = photoInfo.copyright {
                self.copyrightLabel.text = "Copyright
\ (copyright) "
            } else {
                self.copyrightLabel.isHidden = true
            }
            self.nasaImage.image = image
        }
    }
}

```


<pre> } } task.resume() } </pre>
<p>Code description and output</p> <p>The addition from the previous ViewController class is the updateUI function, which brings the ability to update the UI with a photo that's grabbed from the URL query.</p>
<p>Code explanation</p> <p>The updateUI function essentially goes through what PhotoInfoController does, just a little deeper. Using the already decoded photoInfo object, it re-uses the URL to provide itself with an image for the UIImage object we have in our storyboard. Once it makes sure that the data received isn't nil, it sets the title equal to photoInfo.title, description to photoInfo.description, and if the image has a copyright, photoInfo.copyright. It also sets a new UIImage to the image received.</p>

8. Add the funcWithHTTPS function inside the URL extension in URL+Helpers.swift. Replace the statement to retrieve the photo's URL in the updateUI function inside ViewController.swift. Describe the general purpose of the code and its output. Explain how the code works and focus on the concepts we discussed in class (e.g., optionals, enumeration, guard, protocols, and closures). You do not need to explain code that has already been discussed in previous questions. The changes in the code are highlighted in yellow. Feel free to read pp. 895 - 898 from the ADS for reference.

<p>URL+Helpers.swift</p> <pre> func withHTTPS() -> URL? { var components = URLComponents(url: self, resolvingAgainstBaseURL: true) components?.scheme = "https" return components?.url } </pre>
<p>ViewController.swift</p> <pre> func updateUI(with photoInfo: PhotoInfo) { let task = URLSession.shared.dataTask(with: photoInfo.url.withHTTPS()) { (data, </pre>



```

        response, error) in
            guard let data = data,
                  let image = UIImage(data: data) else { return }
            print(image)
            DispatchQueue.main.async {
                self.title = photoInfo.title
                self.descriptionLabel.text =
photoInfo.description
                if let copyright = photoInfo.copyright {
                    self.copyrightLabel.text = "Copyright
\\(copyright)"
                } else {
                    self.copyrightLabel.isHidden = true
                }
                self.nasaImage.image = image
            }
        }
        task.resume()
    }
}

```

Code description and output

Code explanation

Part 3. Apply (15 min)

Start time: _____

9. Modify the application by adding a text field and a button. The text field can accept a date that will be used to load the corresponding photo of the day. The button will initiate the process of retrieving the information and updating the image, description, and copyright information. Assume that the text provided by the user will always be a valid date string (e.g., 2020-11-02).

```

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.
    descriptionLabel.text = ""
    copyrightLabel.text = ""
    photoInfoController.fetchPhotoInfo { (photoInfo) in
        guard let photoInfo = photoInfo else { return }
    }
}

```



```
        self.updateUI(with: photoInfo)
    }

    func updateUI(with photoInfo: PhotoInfo) {
        let task = URLSession.shared.dataTask(with:
photoInfo.url.withHTTPS()) { (data,
        response, error) in
            guard let data = data,
                let image = UIImage(data: data) else {
return }

            print(image)
            DispatchQueue.main.async {
                self.title = photoInfo.title
                self.descriptionLabel.text =
photoInfo.description
                if let copyright = photoInfo.copyright {
                    self.copyrightLabel.text = "Copyright
\ (copyright) "

                } else {
                    self.copyrightLabel.isHidden = true
                }
                self.nasaImage.image = image
            }
        }
        task.resume()
    }
}
```

Reflector questions

1. What was the most useful thing you learned during this session?

2. What did the team do well?



3. What were the challenges that the team encountered?

4. What do you suggest the team do in the next meeting to do better?

5. Rate your team according to the rubric below

Self-rating:

Criteria	Score
Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0

POGIL Activity 13.1: Error Handling

In this activity we will learn about error handling. Specifically we will design code that makes it easy to throw and catch errors while keeping our components decoupled. This design results in code that is easier to extend and adapt.

Please fill in the roles for each member of your team. Take a look at the description of each role to see its responsibilities. In case there are only three people in the group, please assign the same person to the **Presenter** and **Reflector** role. It is a good idea to select roles that you have not recently taken.

Team name: _____

Date: _____

Role	Team Member Name
Manager. Keeps track of time and makes sure everyone contributes appropriately.	Kevin La
Presenter. Talks to the facilitator and other teams.	Ismail Hasan
Reflector. Considers how the team could work and learn more effectively.	Fahad Alsowaylim
Recorder. Records all answers and questions and makes the necessary submission.	Ismail Hasan



POGIL Template by Paul Salvador Inventado is licensed under a Creative Commons

Attribution-NonCommercial-ShareAlike 4.0 International License. Based on a work at <http://bit.ly/2LtrlKv>.

Part 1. Explore (10 min)

Start time: _____

Consider the code below to answer the questions that follow.

```
struct Item {
    var price: Int
    var count: Int
}

class VendingMachine {
    var inventory = [
        "Candy Bar": Item(price: 12, count: 7),
        "Chips": Item(price: 10, count: 4),
        "Pretzels": Item(price: 7, count: 11)
    ]
    var coinsDeposited = 0

    func vend(itemNamed name: String) {
        let item = inventory[name]!

        coinsDeposited -= item.price

        var newItem = item
        newItem.count -= 1
        inventory[name] = newItem

        print("Dispensing \(name)")
    }
}
```

1. What are possible bugs/errors that developers might encounter when they use the VendingMachine class and call its vend function? List as many as you can.

- The itemNamed parameter that is used when vend is called may not be one of the inventory items that's predetermined (e.g. Cookies), it'll crash
- coinsDeposited will be able to go into the negative
- The count of items in the inventory can go into the negative

2. What do you think is a good way to identify those errors and inform the user/developer about them?

The best way to inform a developer about issues in code would simply be to use try/catch statements in the code, if we're aware of them. The developer can pick up on what the try/catch statement is attempting to do, and go along with that.

3. If we were to implement a mobile app. Would it be easy to modify your solution from #2 to show the errors on the user interface? Why or why not?

Try catch statements shouldn't be too difficult to implement into a user interface of a mobile app. After all, the code is run throughout the app itself, so showcasing it wouldn't be a big deal. Doing so also allows the user to know what types of bugs/issues are in the code, not just the developer.

Part 2. Invent (5 min)

Start time: _____

4. What errors would you need to throw based on your answers to #1?

invalidName
insufficientCoins
outOfStock

5. How many catch blocks would you need to catch each error you identified in #4?

3

Part 3. Apply (15 min)

Start time: _____

6. Modify the VendingMachine class' vend function to throw the errors you identified in #4. Feel free to use the enum provided for you below.

```
enum VendingMachineError: Error {
    case invalidSelection
    case insufficientFunds
    case outOfStock
}

func vend(itemNamed name: String) throws {
    guard let item = inventory[name] else {
        throw VendingMachineError.invalidSelection
    }

    guard item.price <= coinsDeposited else {
        throw VendingMachineError.insufficientFunds
    }

    guard item.count > 0 else {
        throw VendingMachineError.outOfStock
    }

    coinsDeposited -= item.price

    var newItem = item
    newItem.count -= 1
    inventory[name] = newItem

    print("Dispensing \(name)")
}
```

7. Create a do-catch block that would catch each error when the vend function is called.

```
do {
    try machine.vend(itemNamed: "Cookies")
} catch VendingMachineError.invalidSelection {
    print("This item doesn't exist in the vending machine.")
} catch VendingMachineError.insufficientFunds {
```



```

    print("You don't have enough coins to make this purchase")
} catch VendingMachineError.outOfStock {
    print("There isn't anymore of that left!")
}

```

Reflector questions

1. What was the most useful thing you learned during this session?

Do { try catch }

2. What did the team do well?

Crack jokes

3. What were the challenges that the team encountered?

none

4. What do you suggest the team do in the next meeting to do better?

There isn't a next meeting :(

5. Rate your team according to the rubric below

Self-rating: 10

Criteria	Score
----------	-------



Answered all problems in the worksheet	10
Partially answered the problems in the worksheet	8
Did not answer the worksheet	0

