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

```cpp
#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

```swift
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

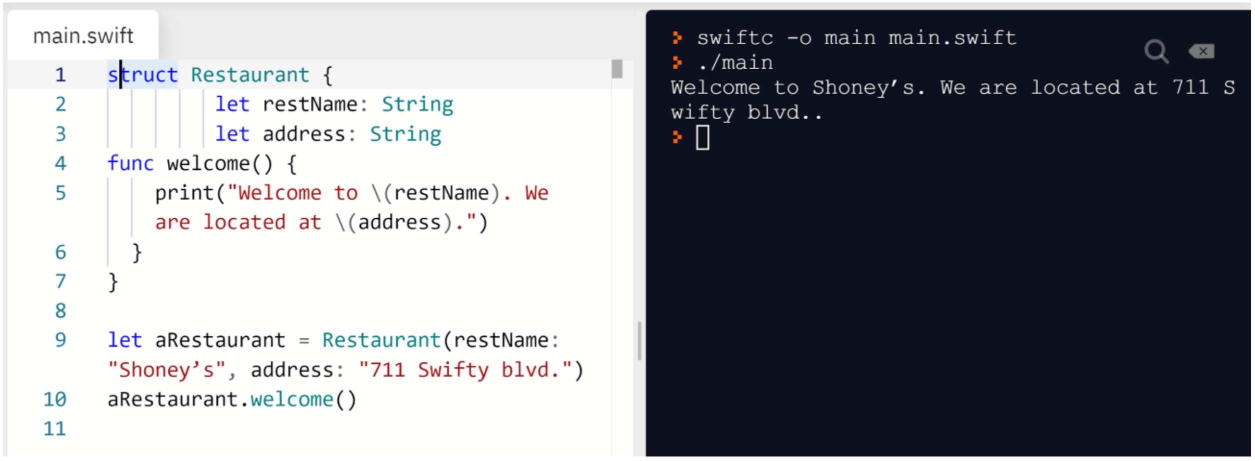
Xcode

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

No Xcode

a. Use an online compiler that supports Swift. For example, repl.it or CS50 sandbox. You may need to create an account to access these websites.
b. Copy your answer in #3 into the editor.

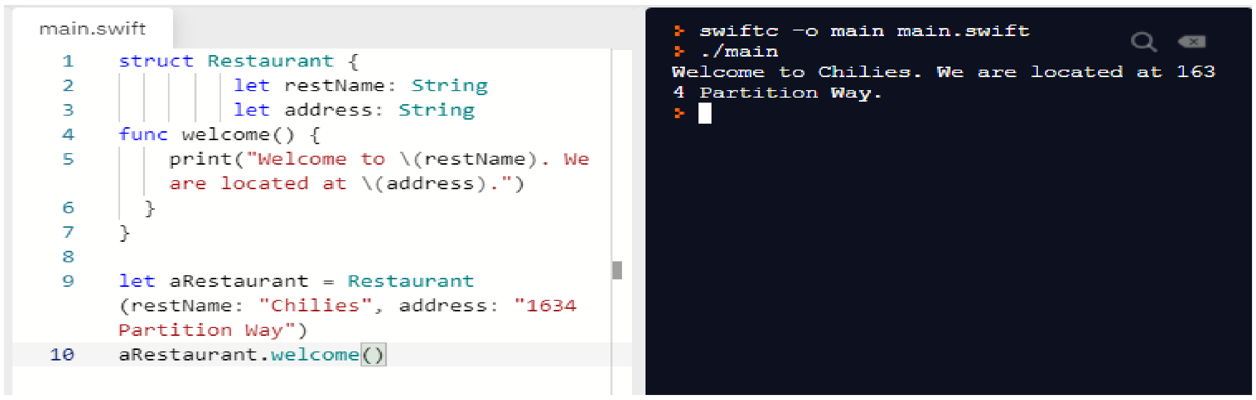
Each member of the group should take a screenshot of their Xcode playground or online compiler showing that the program worked and its output. Place each member's screenshot below.



```
struct Restaurant {
            let restName: String
            let 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()
```

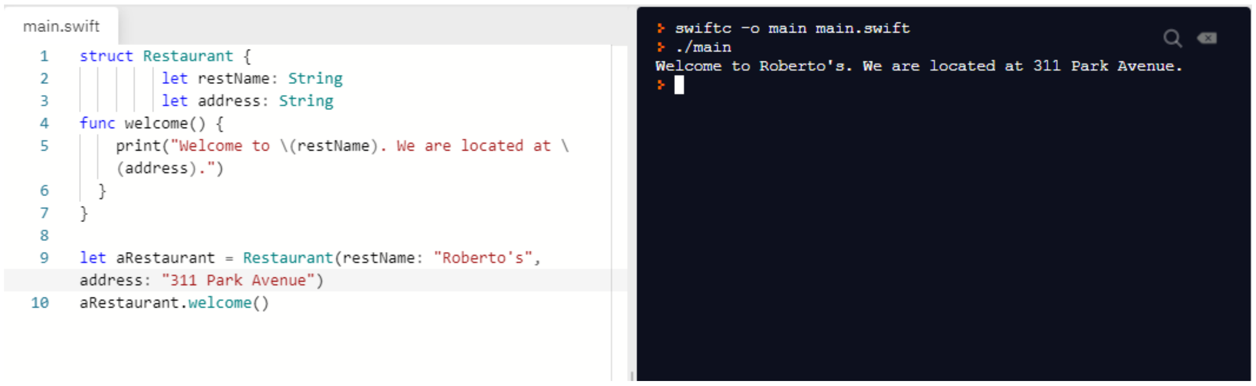
```
> swiftc -o main main.swift
> ./main
Welcome to Shoney's. We are located at 711 S
wifty blvd..
> 
```


main.swift

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

let aRestaurant = Restaurant
(restName: "Chilies", address: "1634
Partition Way")
aRestaurant.welcome()
```

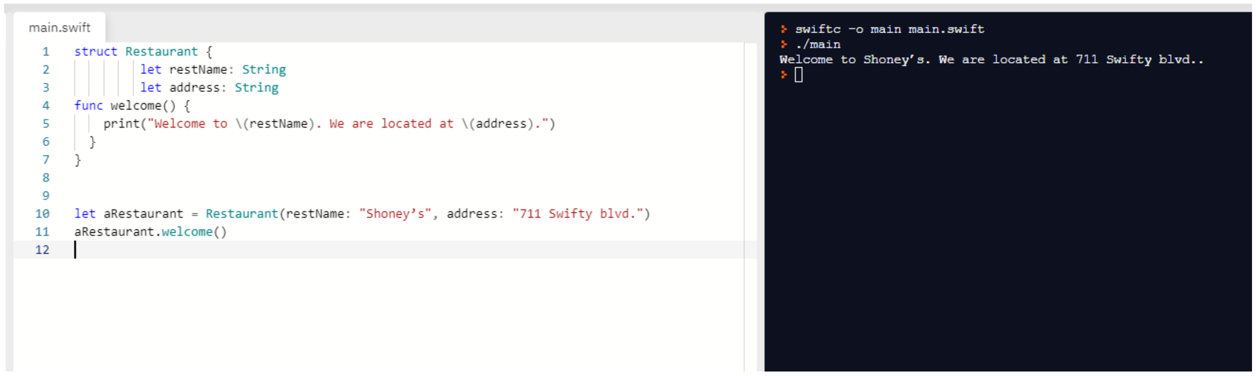
```
> swiftc -o main main.swift
> ./main
Welcome to Chilies. We are located at 163
4 Partition Way.
>
```


main.swift

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

let aRestaurant = Restaurant(restName: "Roberto's",
address: "311 Park Avenue")
aRestaurant.welcome()
```

```
> swiftc -o main main.swift
> ./main
Welcome to Roberto's. We are located at 311 Park Avenue.
>
```

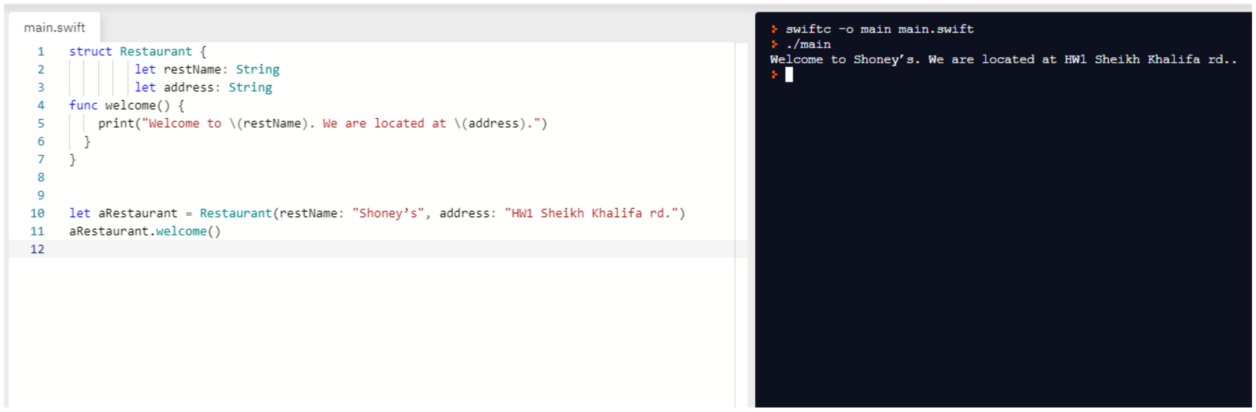

main.swift

```swift
struct Restaurant {
        let restName: String
        let 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()

```

```
> swiftc -o main main.swift
> ./main
Welcome to Shoney’s. We are located at 711 Swifty blvd..
>
```


main.swift

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


let aRestaurant = Restaurant(restName: "Shoney’s", address: "HW1 Sheikh Khalifa rd.")
aRestaurant.welcome()

```

```
> swiftc -o main main.swift
> ./main
Welcome to Shoney’s. We are located at HW1 Sheikh Khalifa rd..
>
```

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

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

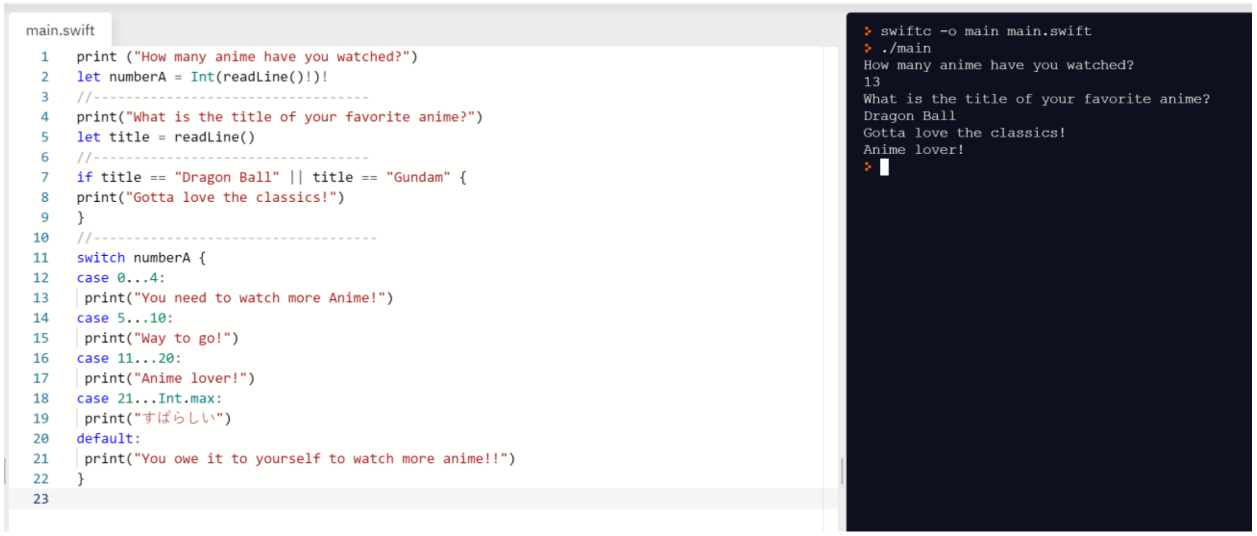
Swift code:
```
print ("How many anime have you watched?")
let numberA = Int(readLine()!)!

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]()). Take a screenshot of the output and place it in the box below.



main.swift

```
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 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!!")
}
```

```
> 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

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)
```

```
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?

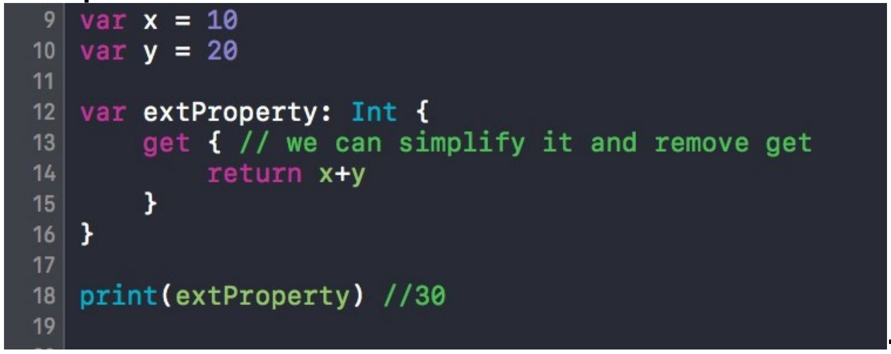
We can utilize various program structure concepts to resolve the program having to be ran multiple times. For example, running the program in a loop and would repeat the number of times desired.

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:



```
var x = 10
var y = 20

var extProperty: Int {
    get { // we can simplify it and remove get
        return x+y
    }
}

print(extProperty) //30
```

-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 internalSteakTempurature 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 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**

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___

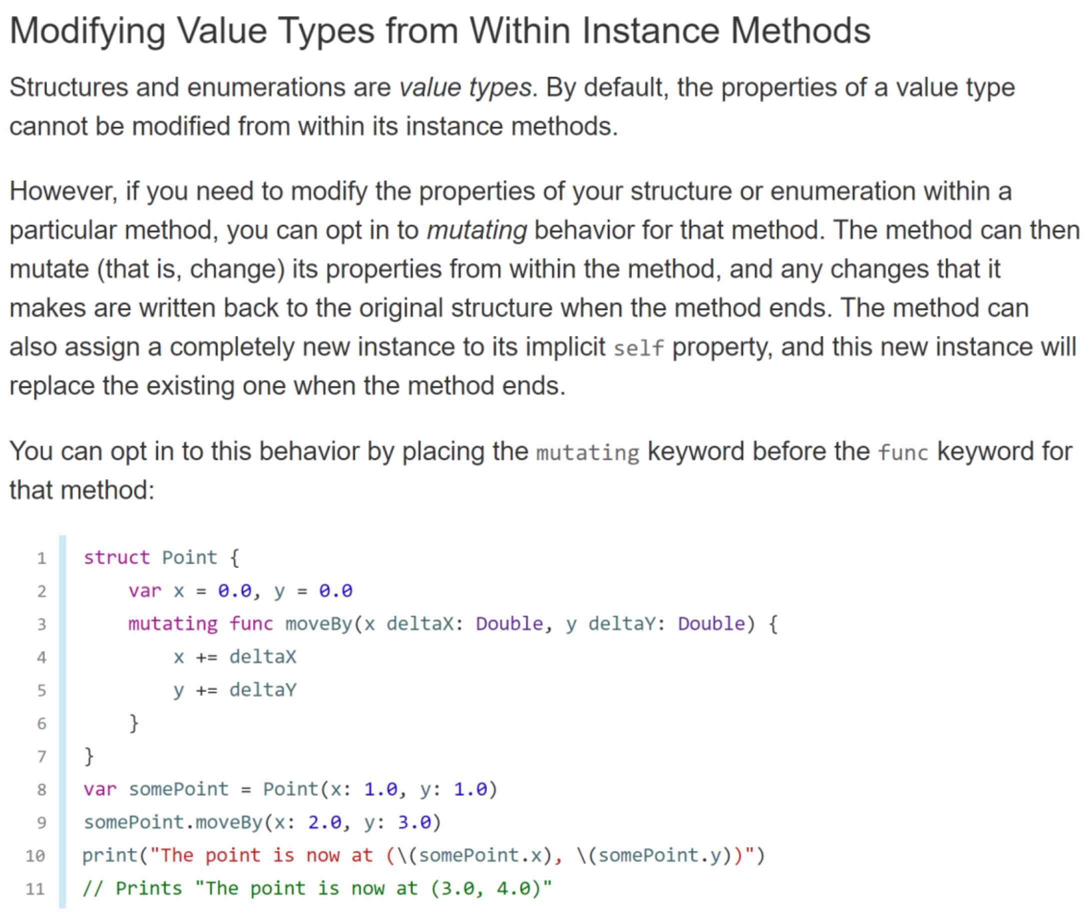
4. Read on structure initializers in the ADS book. How can you use initializers to set the default values of the oven's properties? Create an initializer for your Oven structure. Provide the code below.

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:

```swift
struct Point {
    var x = 0.0, y = 0.0
    mutating func moveBy(x deltaX: Double, y deltaY: Double) {
        x += deltaX
        y += deltaY
    }
}
var somePoint = Point(x: 1.0, y: 1.0)
somePoint.moveBy(x: 2.0, y: 3.0)
print("The point is now at (\(somePoint.x), \(somePoint.y))")
// 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_

6. 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.

```
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	**ExperiencedHiker**	**ExpertHiker**
totalSteps	totalSteps totalClimbs	totalSteps totalClimbs totalCamps
walk stats	walk climb stats	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____

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.
   a. It has a property called number
   b. It has a property observer for number that ensures number's value is always from 1 to 12. Otherwise, it keeps its old value.
   c. It provides a computed property called name that returns the corresponding month name. For example, 1 is January, 2 is February, and so forth.
   d. It has a custom initializer that does not accept any parameters, but sets the number to 1.
   e. 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 XCTests.

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____________________________ Date: ___09/22/2020___________

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____

3. 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 = myDictionaryt.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.

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**

`// Loop 1`
`for index in 1...10 {`
`    if index % 2 == 0 {`
`        print(index)`
`    }`
`}`
	`2`
	`4`
	`6`
	`8`
	`10`

`// Loop 2`
`let expenses = [3.99, 10.58, 25.00, 15.50]`
 	
`for expense in expenses {`
`    print(expense)`
`}`
	`3.99`
	`10.58`
	`25.00`
	`15.50`

`// Loop 3`
`let expenses = [3.99, 10.58, 25.00, 15.50]`
 	
`for (index,expense) in`
`expenses.enumerated() {`
`    print("\(index): \(expense)")`
`}`
	`0: 3.99`
	`1: 10.58`
	`2: 25.0`
	`3: 15.5`

`// Loop 4`
 	
`let grades = ["A":90, "B": 80, "C": 70,`
`              "D": 60, "F" :0]`
 	
`for (letter, grade) in grades {`
`  print("\(letter) >\(grade)")`
`}`
	`A >90`
	`C >70`
	`B >80`
	`D >60`
	`F >0`

Part 2. Invent (15 min) Start time: _1:42_____

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

Loop 1

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__

3. 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