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: ______

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

Map

```
let names = ["Johnny", "Nellie", "Aaron", "Rachel"]

// Creates a new array of full names by adding "Smith" to each
// first name
let fullNames = names.map { (name) -> String in
    return name + " Smith"
}
```

Filter

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

print(numbersLessThan20)
```

Reduce

```
let numbers = [8, 6, 7, 5, 3, 0, 9]
let total = numbers.reduce(0) { (currentTotal, newValue) ->
Int in
    return currentTotal + newValue
}
```

Part 3. Apply (30 min) Start time: ______

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 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 productPrices = products.map {
        (product) -> Double in
        return product.totalPrice
    }
    let filteredPrices = productPrices.filter {
        (price) -> Bool in
        return price > 0.5
    }
    let total = filteredPrices.reduce(0) {
        (currentTotal, newValue) -> Double in
        return currentTotal + newValue
    }
    return total
}
```

```
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.

```
// place your code here
func displayProducts(products: [Product], displayFunction:
(String, Double, Int) -> Void) -> Void {
    for product in products {
        displayFunction(product.name, product.price,
                        product.quantity)
    }
}
```

```
// 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?

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