Swift Assignment #4
Functions and Closures

CS275 Fall 2021
15 points
due Tuesday, Oct. 5th, 11:59 pm

1 Functions and Closures

You'll do more practice with second-order functions.

1.1 Example functions

Here's an example of what to do. Suppose I want a function that will take an array of integers and return an array of just the positive integers from the original array. Then I could give you this skeleton:

```
func positivesOnly(_ intArr: [Int]) -> [Int] {
    let r = // fill this in with a single statement using filter
    return r
}
```

And I would call the function this way:

```
let ints = [5, -9, 0, 23, 2, -7, 5, -1]
print("positives are: \(positivesOnly(ints))")
```

And the result would be:

```
positives are:  [5, 23, 2, 5]
```

For this example, here's the code that I would be looking for:

```
func positivesOnly(_ intArr: [Int]) -> [Int] {
    let r = intArr.filter { $0 > 0 }
    return r
}
```

In other words, just a single call using a higher-order function and an inline closure.

Now, do the following. For each function, I show an example call, and the output you should get, in cyan.

(1) Write this function:

```
// convert all characters in all strings in the input array to lower case
func toLower(_ stringArr: [String]) -> [String] {
    let s = // fill in this line with a single statement using map(_:)
    return s
```

```
}

let s1 = ["Now", "is", "THE", "tImE"]
print(toLower(s1))
["now", "is", "the", "time"]
```

(2) Write this function:

```
// add together each of the strings in the input array that can be converted
// to an integer and return the sum
// use map(_:) and reduce(_:_:)
func addInts(_ stringArr: [String]) -> Int {
    let s = // fill this in with a single statement using map(_:)
    let rtnval = // fill this in with a single statement using reduce(_:_:)
    return rtnval
}

let s2 = ["0", "one", "2", "three", "4ff", "5", "six", "Seven", "8", "0"]
print(addInts(s2))
15
```

(3) Write this function:

```
// add the absolute values of the entries in intArr and return the sum
func addMagnitudes(_ intArr: [Int]) -> Int {
    let s = // fill this in with a single statement using map(_:)
    let rtnval = // fill this in with a single statement using reduce(_:_:)
    return rtnval
}

let s3 = [0, 1, -1, 2, -2, 3, -3, 4, -4]
print(addMagnitudes(s3))
20
```

(4) Write this function:

```
// add together the positive values from the input array and return the sum
func addPositives(_ intArr: [Int]) -> Int {
    let s = // fill this in with a single statement using filter(_:)
    let rtnval = // fill this in with a single statement using reduce(_:_:)
    return rtnval
}

let s4 = [0, 1, -1, 2, -2, 3, -3, 4, -4]
print(addPositives(s4))
10
```

(5) Write this function:

```
// concatenate the entries--without a space--in the input array and return the result
func concat(_ stringArr: [String]) -> String {
    let s = // fill this in with a single statement using reduce(_:_:)
    return s
}
```

```
let s5 = ["a", "man", "a", "plan", "a", "canal", "panama"]
print(smashTogether(s5))
amanaplanacanalpanama
```

(6) Write this function:

```
// count the number of even integers and odd integers
// in the input array return the result as a named tuple
func evensOdds(_ intArr: [Int]) -> (evens: Int, odds: Int) {
    // here, I'm not specifying how to write the function
    // but you do need to use reduce(_:_:)
    // and you can choose to use an auxiliary function instead of an inline closure
}

let s6 = 0...10
print(evensOdds(Array(s6)))
(evens:  6, odds:  5)
```

(7) Use this function:

```
func inSet(set: Set<String>) -> (String) -> Bool {
    func f(_ e: String) -> Bool {
        return set.contains(e)
    }
    return f
}
```

and write these two functions:

```
func listIntersection(_ setOne: Set<String>, with checker: (String) -> Bool) -> Set<String> {
    let rtnval = // fill this in with a single statement using filter(_:)
    return rtnval
}

func listDifference(_ setOne: Set<String>, with checker: (String) -> Bool) -> Set<String> {
    let rtnval = // fill this in with a single statement using filter(_:)
    return rtnval
}

let myGroceries: Set = ["bugles", "beer", "carrots", "tapenade", "black licorice"]
let wifeGroceries: Set = ["carrots", "bugles", "mushrooms", "mayonnaise"]
print(listIntersection(myGroceries, with: inSet(set: wifeGroceries)))
print(listDifference(myGroceries, with: inSet(set: wifeGroceries)))
["carrots", "bugles"]
["tapenade", "black licorice", "beer"]
```

2 Graduate Students

Graduate students, and undergraduates who would like a bit of extra credit:

(1) Rewrite `evensOdds(_:)` to use an inline closure.

(2) Write these two functions:

```
func makeSieve(for val: Int) -> (Int) -> Bool
func sieveOfEratosthenes(_ n: Int, makeSieve: (Int) -> (Int) -> Bool) -> [Int]
```

which I can use in this way:

```
print(sieveOfEratosthenes(100, makeSieve: makeSieve))
[1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83,
89, 97]
```

Use `filter(_:)` somewhere—the Sieve of E is fundamentally a repeated filtering operation.

3 What to Submit

Do all of these in a single Xcode Playground. Submit just your Swift file. You can see your .swift file in your Playground directory, with the name `Contents.swift`. Rename it `assignment-four.netid.swift`, using your netid.