**COMP 333**

**Summer 2021**

**Extensions and Protocols in Swift**

1.) Use extension to add an add method to Int, which takes another Int and returns the sum of the two Ints. An example call is below:

5.add(6) // returns 11

extension Int {

func add(\_ other: Int) -> Int {

return self + other;

}

}

2.) Define a protocol named Equality, which defines an equals method. equals takes something of the same type it is called on, and returns a Bool indicating whether the two values equal each other or not. As a hint, the type Self refers to whatever type it was called on. Example calls are below (assuming an extension is defined elsewhere adding equals to Int):

5.equals(5) // returns true

5.equals(6) // returns false

5.equals("foo") // compile-time error; Int and String are not

// not the same type

protocol Equality {

func equals(\_ other: Self) -> Bool;

}

3.) Use extension to say that Int satisfies the Equality protocol you defined above. This adds the equals method to Int. As a hint, == is used to test if two Ints are equal or not.

extension Int : Equality {

func equals(\_ other: Int) -> Bool {

return self == other;

}

}

4.) Consider the following enum definition:

indirect enum List<A> {

case cons(A, List<A>)

case empty

}

Define an extension which will add an evens method specifically to List<Int>, where evens returns a list of all the even numbers in the input list. As a hint, this works in a manner similar to filter. Example calls are below:

let list1 = List.cons(2, List.cons(3, List.cons(4, List.empty)))

let list2 = List.cons("foo", List.cons("bar", List.empty))

list1.evens() // returns List.cons(2, List.cons(4, List.empty))

list2.evens() // compile-time error; evens() is only available

// on List<Int> and list2 is of type List<String>

extension List where A == Int {

func evens() -> List<Int> {

switch self {

case .empty:

return List.empty;

case .cons(let head, let tail):

let rest = tail.evens();

if head % 2 == 0 {

return List.cons(head, rest);

} else {

return rest;

}

}

}

}