**COMP 333**

**Summer 2021**

**Generics, Parametric Polymorphism, and Higher-Order Functions in Swift**

1.) Define a function that takes a value of some generic type A, and returns the same value.

func myFunc<A>(a: A) -> A {

return a;

}

2.) Define a function that takes values of generic types A and B, and returns a pair of these values.

func myFunc<A, B>(a: A, b: B) -> (A, B) {

return (a, b)

}

3.) Write the body of the following Swift function. As a hint, only one possible body (which typechecks) exists.

func myFunc<A, B, C, D>(a: A, b: B,

f1: (A) -> C,

f2: (A, C) -> D) -> (C, D) {

let c = f1(a);

let d = f2(a, c);

return (c, d);

}

4.) Consider the following enum definition, defining the structure of a linked list:

indirect enum List<A> {

case cons(A, List<A>)

case empty

}

4.a.) Define the map function, which has the following signature:

func map<A, B>(list: List<A>, f: (A) -> B) -> List<B> {

switch list {

case .empty:

return List.empty;

case .cons(let head, let tail):

return List.cons(f(head), map(list: tail, f: f))

}

}

4.b.) Define the foldLeft function, which has the following signature:

func foldLeft<A, B>(list: List<A>,

accum: B,

f: (B, A) -> B) -> B {

switch list {

case .empty:

return accum;

case .cons(let head, let tail):

return foldLeft(list: tail,

accum: f(accum, head),

f: f)

}

}