COMP 333
Fall 2021

Algebraic Datatypes and Pattern Matching in Swift

1.) Define an `enum` named `MyBool` which represents truth and falsehood.

```
enum MyBool {
  case truth
  case falsehood
}
```

2.) Define an `enum` named `MyList` which encodes a singly-linked list of integers, using the same cons/nil structure that we used in assignment 1.

```
indirect enum MyList {
  case cons(Int, MyList)
  case empty // nil is already used in Swift elsewhere,
             // so a different name must be used
}
```

3.) Using the prior enum definition, create a list containing 1, 2, and 3, in that order.

```
MyList.cons(1, MyList.cons(2, MyList.cons(3, MyList.empty)))
```

4.) Write a `switch` which will pattern match on a variable named `list`, and do one of the following:

- If the list starts with a 2, return 0
- If the list starts with a 3, followed by a 4, return 1
- For any other non-empty list, return the value of the first element
- If the list is empty, return -1

```
switch list {
  case .cons(2, _):
    return 0
  case .cons(3, .cons(4, _)):
    return 1
  case .cons(let first, _):
    return first
  case .empty:
    return -1
}
```

5.) Write a function named `length` which takes a list as a parameter, and recursively computes the length of the given list.

```
func length(list: MyList) -> Int {
  switch list {
    case .empty:
      return 0
    case .cons(_, let tail):
      return 1 + length(list: tail)
  }
}
```