Problem Wk.2.3.3: Inheritance and State Machines

Recall that we have defined a Python class `sm.SM` to represent state machines. Here we consider a special type of state machine, whose states are always integers that start at 0 and increment by 1 on each transition. We can represent this new type of state machines as a Python subclass of `sm.SM` called `CountingStateMachine`.

We wish to use the `CountingStateMachine` class to define new subclasses that each provide a single new method `getOutput(self, state, inp)` which returns *just the output* for that state and input; the `CountingStateMachine` will take care of managing and incrementing the state, so its subclasses don't have to worry about it.

Here is an example of a subclass of `CountingStateMachine`.

```
class CountMod5(CountingStateMachine):
    def getOutput(self, state, inp):
        return state % 5
```

Instances of `CountMod5` generate output sequences of the form `0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, ....`

Define the `CountingStateMachine` class. Since `CountingStateMachine` is a subclass of `sm.SM`, you will have to provide definitions of the `startState` instance variable and `getNextValues` method, just as we have done for other state machines. You can assume that every subclass of `CountingStateMachine` will provide an appropriate `getOutput` method; `getNextValues` should use `getOutput` to produce the output for a given `state` and `inp`.

Then, define a subclass of `CountingStateMachine` called `AlternateZeros`. Instances of `AlternateZeros` should be state machines for which, on even steps, the output is the same as the input, and on odd steps, the output is 0. That is, given inputs, $i_0, i_1, i_2, i_3, \ldots$, they generate outputs, $i_0, 0, i_2, 0, \ldots$

Hint: Study the `Staff601` and `SM` classes in Lecture 2. Note that a superclass method can refer to attributes and methods of the subclasses.

6.01SC Introduction to Electrical Engineering and Computer Science

Spring 2011