Problem Wk.4.4.1: LTISM

We have seen how to build up complex state machines out of primitives, and showed that we could run them. The only problem with that way of working is that we never have an explicit representation of the entire system as a difference equation. Later, we will automatically compute difference equations for systems created with combinators; for now, we'll create state machines directly from the coefficients for a difference equation.

For each of these systems to be a well-defined state machine, we have to specify the initial state, which can be characterized by values for k previous output values and j previous input values. Recall that a difference equation is in the form:

$$y[n] = c_0 y[n-1] + c_1 y[n-2] + \ldots + c_{k-1} y[n-k] + d_0 x[n] + d_1 x[n-1] + \ldots + d_j x[n-j]$$

We have defined a special class of state machines, called `LTISM`, that can be used to easily implement any machine in this class. The initializer is

```
LTISM(dCoeffs, cCoeffs, previousInputs, previousOutputs)
```

where

- `dCoeffs` is a list of the coefficients d_0 through d_j, in that order.
- `cCoeffs` is a list of the coefficients c_0 through c_{k-1}, in that order.
- `previousInputs` is a list of $x[-1]$ through $x[-j]$, in that order.
- `previousOutputs` is a list of $y[-1]$ through $y[-k]$, in that order.

Here are Python expressions that construct instances of `LTISM`:

- Output at time `n` is 3 times the input at time `n`:

  ```
  # y[n] = 3 x[n]
  LTISM([3],[])
  ```

- Output at time `n` is the input at time `n-1` (this is our old friend the delay machine):

  ```
  # y[n] = x[n-1]
  LTISM([0, 1],[], [0], [])
  ```

  Here, we've specified that the $x[-1] = 0$.
- Output at time `n` is the 2 times the input at time `n-2`:

  ```
  # y[n] = 2 x[n-2]
  LTISM([0, 0, 2],[], [4, 10], [])
  ```

  Here, we've specified that the $x[-1] = 4$ and $x[-2] = 10$.
- Output at time `n` is the 2 times the output at time `n-1`:

  ```
  # y[n] = 2 y[n-1]
  LTISM([],[2], [], [1])
  ```

  Here, we've specified that the $y[-1] = 1$. (What would happen if we set it to 0?)
- Output at time `n` is the output at time `n-1` plus the output at time `n-2`:

  ```
  # y[n] = y[n-1] + y[n-2]
  LTISM([],[1, 1], [], [1, 1])
  ```

  Here, we've specified that the $y[-1] = 1$ and $y[-2] = 1$. This formula generates a familiar sequence. Do you recognize it?

Show how to construct the following state machines as an instance of the `LTISM` class. Below, you will need to enter the elements of the four lists that are needed to create an instance:

```
LTISM(dCoeffs, cCoeffs, previousInputs, previousOutputs)
```

If any argument list is empty, enter `none`, otherwise enter a sequence of numbers separated by spaces (no commas, parens, brackets, etc).

1. A state machine whose output at time `n` is the sum of its inputs up to and including time `n`. Assume $y[-1] = 10$.
   dCoeffs (input):
   cCoeffs (output):
   previousInputs:
   previousOutputs:
2. A state machine whose output at time `n` is the sum of its inputs up to and including time `n-1`. Assume $y[-1] = 10$ and $x[-1] = 0$.
   dCoeffs (input):
   cCoeffs (output):
   previousInputs:
   previousOutputs:
3. A state machine whose output at time `n` is the sum of its inputs up to and including time `n-1` each scaled by 0.1. Assume $y[-1] = 1$ and $x[-1] = 0$.
   dCoeffs (input):
   cCoeffs (output):
   previousInputs:
   previousOutputs:

6.01SC Introduction to Electrical Engineering and Computer Science

Spring 2011