Problem Wk.3.1.4: Combining accounts

Consider two different kinds of bank accounts:

- `BA1`: Fee of $100 on every (non-zero) deposit and withdrawal; 2% interest per time step.

```
class BA1(sm.SM):
    startState = 0
    def getNextValues(self, state, inp):
        if inp != 0:
            newState = state * 1.02 + inp - 100
        else:
            newState = state * 1.02
        return (newState, newState)
```

- `BA2`: No transaction fee; 1% interest per time step.

```
class BA2(sm.SM):
    startState = 0
    def getNextValues(self, state, inp):
        newState = state * 1.01 + inp
        return (newState, newState)
```

Part 1: Maximize

Make a state machine that computes the balances of both types of accounts, but whose output is the maximum of the two balances.

The input to the state machine is a number (positive numbers are deposits, negative numbers are withdrawals); the output is a number that represents the maximum of the two balances.

Start by constructing a state machine whose input is a number and whose output is a tuple with two balances: the balance an account of the first type would have had with the input and the balance an account of the second type would have had with the input. Then combine this machine with an instance of a `sm.PureFunction` (defined in an earlier problem) to produce the desired composite machine.

The definition of `sm.PureFunction` is already included in the problem. Combinators (as defined in Section 4.2 of the notes) need to be accessed from the `sm` module, that is, `sm.Cascade`, `sm.Parallel`, `sm.Parallel2`, etc.

Do not define any new state-machine subclasses. You can do this all with combinators and `sm.PureFunction`.

Assign your machine to the name `maxAccount`.

```
ba1 = BA1()
ba2 = BA2()
maxAccount = None
```

Part 2: Investment

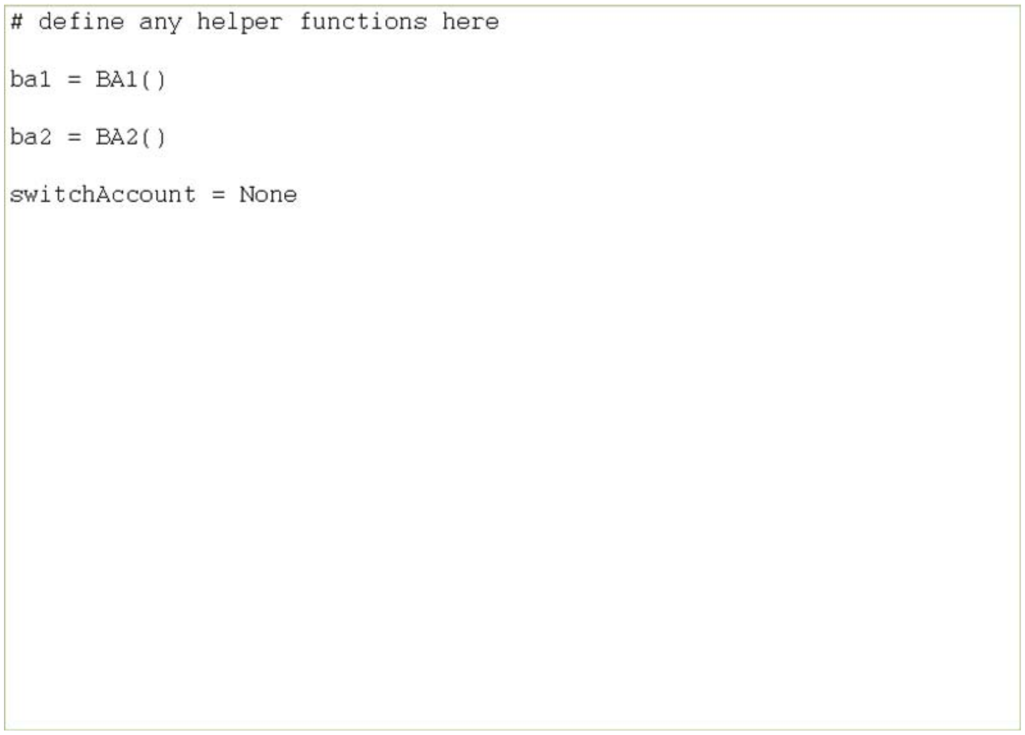
My business has two bank accounts, as above. I put any deposit or withdrawal whose **magnitude** is greater than $3,000 in the account of type 1, and all other deposits and withdrawals in the account of type 2. On every step, both accounts should continue to earn the relevant interest. The output should be the sum of the balances in the two accounts.

Implement this by composing the two bank accounts using `sm.Parallel2` and cascading it with two simple machines you implement using `sm.PureFunction`.

The definition of `sm.PureFunction` is already included in the problem. Combinators need to be accessed from the `sm` module, that is, `sm.Cascade`, `sm.Parallel2`, etc.

Assign your machine to the name `switchAccount`.

You will probably want to use helper functions in your solution.



```
# define any helper functions here

ba1 = BA1()

ba2 = BA2()

switchAccount = None
```