MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.01—Introduction to EECS I
Spring Semester, 2008

NanoQuiz Week #4 (sections 1 and 2)

Name: ________________________ **Athena userid:** __________________`@mit.edu`

This quiz is due promptly 15 minutes after the start of the lab period.

You may use the weekly assignment handout, but the quiz is otherwise closed book and closed computer.

1. The `makeSumSM` procedure returns a state machine whose initial state is 0, and whose output at time `t` is the sum of all the inputs from time 0 through time `t−1`. The `makeIncr` procedure takes an initial value as input and returns a state machine whose initial state is the initial value, and whose output at time `t` is the input at time `t−1` plus 1. (These are both the same as the ones from the software lab).

```
def makeSumSM():
    return PrimitiveSM(lambda s, i: s + i,
                       lambda s: s,
                       lambda : 0)
def makeIncr(init = 0):
    return PrimitiveSM(lambda s, i: i+1,
                       lambda s: s,
                       lambda : init)
```

Now we serially compose these machines so that the output of the `incr` machine is the input of the `sum` machine, and run it:

```
m = SerialSM(makeIncr(0), makeSumSM())
transduce(m, [10, 20, 30, 40])
```

In the table below, fill out the values of the inputs, states, and outputs of the composite machine (where $output_1$ is the same as $input_2$).

step	$input_1$	$state_1$	$output_1$	$state_2$	$output_2$
0					
1					
2					
3					

See other side.

2. Consider the following code:

```
class Party:
    def __init__(self, food):
        self.food = food
        self.guestsAtTheParty = []
    def addFood(self, newFood):
        self.food.append(newFood)
    def welcomeGuest(self, guest):
        self.guestsAtTheParty.append(guest)

class InvitationOnlyParty(Party):
    def __init__(self, food, invitedGuests):
        self.invitedGuests = invitedGuests
        Party.__init__(self, food)
    def welcomeGuest(self, guest):
        if guest in self.invitedGuests:
            Party.welcomeGuest(self, guest)
```

Assume we evaluate the code above, and then type the following expressions into Python in order. Say what Python will print out, in the blank spaces.

(a)
```
> p = Party(['cake', 'iceCream'])
> p.welcomeGuest('Pat')
> p.welcomGuest('Kim')
> p.guestsAtTheParty
```

(b)
```
> p2 = InvitationOnlyParty(['cherries', 'herring'],
                           ['Sydney', 'Pat', 'Michael'])
> p2.welcomeGuest('Pat')
> p2.welcomeGuest('Kim')
> p2.guestsAtTheParty
```