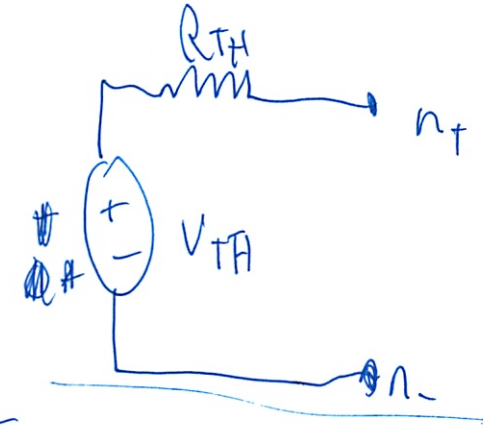
Thevin + Norton Reading Review

12/13

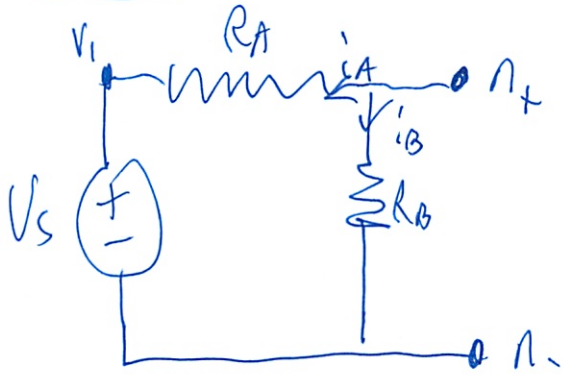
Can't abstract circuits
- but can abstract pieces

1) Get open circuit V across terminals ($I=0$)
2) Then short circuit current



$$i_{sc} = -\frac{V_{TH}}{R_{Th}}$$

Example



So first set n_- to 0 and measure n_+
But first NVCC

$$V_+ - V_1 = i_A R_A$$
$$V_1 - V_- = V_s$$
$$V_+ - V_- = i_B R_B$$
$$-I_A - I_B = 0$$
$$I_A - I_s = 0$$

(2)

Solve

$$V_+ = V_S \frac{R_B}{R_A + R_B}$$

Try on own

$$-\frac{V_+ - V_1}{R_A} = \frac{V_+ - V_-}{R_B} \qquad I_A = I_S$$

$$-R_B(V_+ - V_-) = R_A(V_+ - V_-)$$

↑ can not do when multiple items are really multiplying everything by R_B:

$$R_B(V_+ - V_+) = \frac{R_B(V_+ - V_-)}{R_B}$$

$$-R_B V_+ + R_B V_1 = R_A V_+ - R_A V_-$$

Get $V_1 - V_-$

$$R_A V_+ + R_B V_+ = R_B V_1 + R_A V_- \qquad -V_S = -V_1 + V_-$$

$$V_+(R_A + R_B) = \text{~~that how factor R was copy error~~}$$

↑ how to factor

$$-R_A V_S$$

$$V_+ = -V_S \frac{R_A}{R_A + R_B}$$ ← A not B

↑ negative

I don't get how I screwed this up

$V_- = 0$ – that simplifies this greatly

2b

don't forget

$$-\frac{V_+ - V_1}{R_A} = \frac{V_+}{R_B}$$

$$-R_B(V_+ - V_1) = R_A V_+$$

$$-R_B V_+ + R_B V_1 = R_A V_+$$

$$R_A V_+ + R_B V_+ = R_B V_1$$

$$V_+(R_A + R_B) = R_B V_S$$

$$V_+ = V_S \frac{R_B}{R_A + R_B}$$

✓

(3)

Now need short circuit current

Connect wire n_+ n_-:

$$V_+ - V_1 = i_A R_A$$

$$V_{1-} - V_- = V_S$$

$$V_+ - V_- = i_B R_B$$

$$V_+ = V_-$$

$$i_{SC} = i_A - i_B = 0$$

$$i_A - i_S = 0$$

$$V_- = 0$$

$$i_{SC} = -\frac{V_S}{R_A}$$

So $R_{th} = \dfrac{-V_{th}}{i_{SC}}$

$$= V_S \frac{R_B}{R_A + R_B} \frac{V_S}{R_A}$$

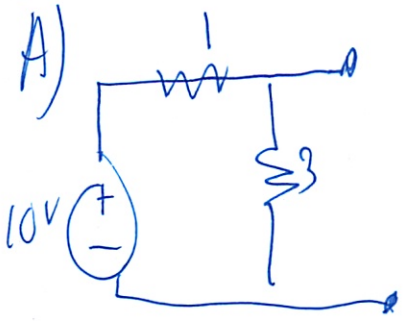
$$= \frac{R_A R_B}{R_A + R_B}$$

Norton - just give i_{SC} - don't convert to V_{TH}

What did we do in review section - was very different w/ real #

(4)

A)



Oh same as book example but w/ #
And he did it an "intuitive" way

#1) What is voltage drop b/w pins
- voltage divider

$\frac{3}{4} \cdot 10V = 7.5$

3 ← below total

4 ← nothing extra added

#2) Short circuit
find eq resistance

$R_{Th} = \frac{V_{Th}}{I_{sc}} =$

$V = IR$

V ← 0

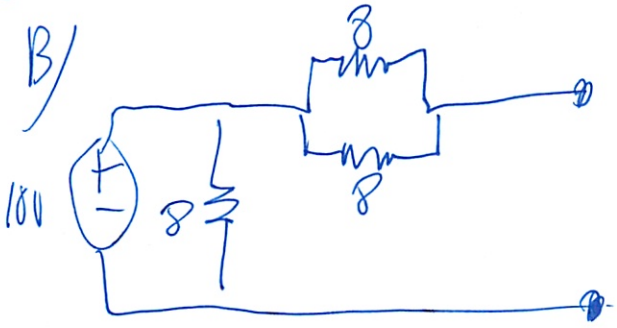
R ← 1 only 1 resistor in use

$I = 10$

So $R_{TH} = \frac{7.5}{10} = .75$

(5)

B/



Let me try on own

First replace  w/ 

1) Voltage

- ?
- ? will all just ~~gone~~ go around loop w/ 8
- ~~so V=0~~

$V_{th} = 10$

- no current drawn - so ignore 4
- 8V not used either
- if did voltage divider $= \frac{8}{8}$

(why am I so bad at this!)

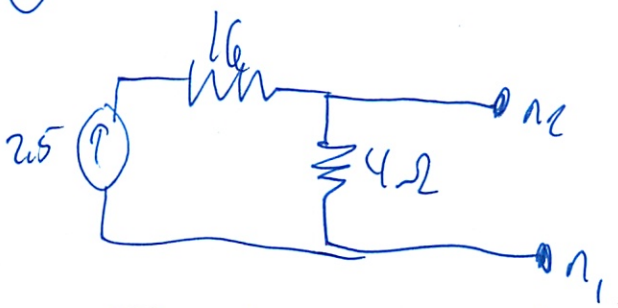
2) Short circuit

$$I_{sc} = \frac{V}{R} = \frac{10}{4} = \frac{5}{2} \quad \checkmark \qquad V=IR$$

↳ the 8 is bypassed again

3) $R_{th} = 4\,\Omega$ ✓

(6)



I should do nVCL if stuck

6.01 Final Exam: Spring 2009

Name:

Practice 12/15

Enter all answers in the boxes provided.
You may use any paper materials you have brought.
No computers, cell phones, or music players.

For staff use:

1.	/14
2.	/14
3.	/14
4.	/14
5.	/15
6.	/14
7.	/15
total:	/100

worksheet

for each variable
if not none
Combine coeff w/ values and sum
end with # + variable # = #
- # / # = variable

1 Programming (14 points)

In the Equation class, we represent a linear equation

$$a_0x_0 + \ldots + a_nx_n = c$$

with a list of coefficients, `coeffs`, corresponding to $[a_0, \ldots, a_n]$ and the `constant`, `c`.

```
class Equation:
    def __init__(self, coeffs, variableNames, constant):
        self.variableNames = variableNames    # List of variable names
        self.coeffs = coeffs                  # List of coefficients
        self.constant = constant              # Constant (right hand side)
```

Suppose that we had a list of values of `[v0, ..., vn]` for the variables $(x_0, \ldots x_n)$ in which one of the values is `None` and all of the rest of the values are numbers.

These numeric values are enough to determine a value for the unspecified variable.

Write a procedure `determineValue` that takes an instance of the `Equation` class and a list of values (exactly one of which is `None`) and returns the value of the unspecified variable.

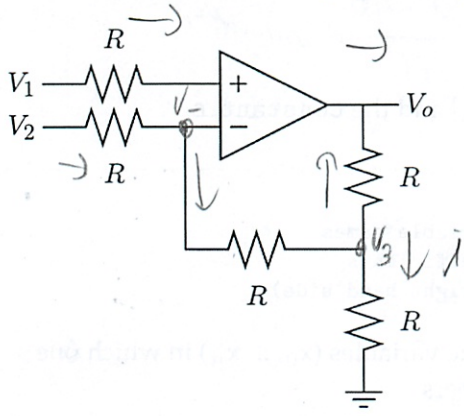
very tough
- Solve a diff eq
- how do you do manually to
- but not polynomial

```
def determineValue(eq, values):
    sum = 0
    unknownPos = None
    i = 0
    for coeff in eq.coeffs:        for i in range(len(eq.coeffs)):
        if values[i] is not None:
            sum = sum + eq.coeffs[i] * values[i]
        else:
            unknownPos = i
    c = eq.constant
    return (c - sum) / eq.coeffs[unknownPos]
```

Oh they start w/ ans = c
then subtract items - could have done that too

2 Circuits (14 points)

Let V_- represent the voltage at the negative input of the op-amp in the following circuit.



Notice that if we knew the values of V_o and V_2 then we could calculate the value of V_-. The resulting value of V_- will be a linear combination of V_2 and V_o of the following form:

$$V_- = \alpha V_o + \beta V_2. \quad (1)$$

Part a. Use Equation 1 to find an expression for the output voltage V_o in terms of the input voltages V_1 and V_2. Which of the following expressions gives the result?

Hint: it is not necessary to determine the values of α and β to do this part of the question.

$V_A = \alpha V_2 - \beta V_1$ $\quad V_B = \dfrac{\alpha V_2 - V_1}{\beta}$ $\quad V_C = \beta V_2 - \alpha V_1$

$V_D = \dfrac{\beta V_2 - V_1}{\alpha}$ $\quad V_E = \dfrac{V_2 - \alpha V_1}{\beta}$ $\quad V_F = \dfrac{V_2 - \beta V_1}{\alpha}$

$V_G = \alpha V_1 - \beta V_2$ $\quad V_H = \dfrac{\alpha V_1 - V_2}{\beta}$ $\quad V_I = \beta V_1 - \alpha V_2$

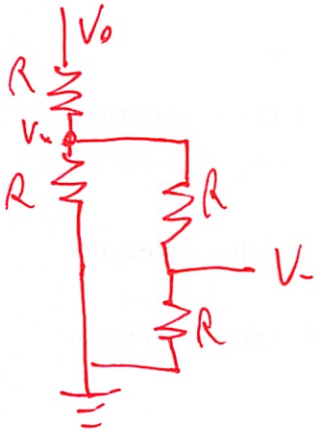
$V_J = \dfrac{\beta V_1 - V_2}{\alpha}$ $\quad V_K = \dfrac{V_1 - \alpha V_2}{\beta}$ $\quad V_L = \dfrac{V_1 - \beta V_2}{\alpha}$

$V_o =$ (V_A or V_B or ... or V_L or none) V_L

Part b. Find α and β in Equation 1 (previous page). **Hint:** Try superposition.

α = 1/5 β = 3/5

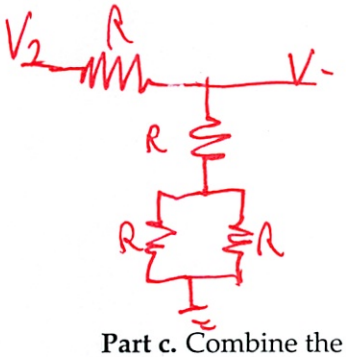
If $V_2 = 0$ $V_- = \alpha V_0$



$V_- = \frac{1}{2} V_x$

$V_x = \frac{2\|R}{R + 2R\|R} V_0 = \frac{\frac{2}{3}R}{R + \frac{2}{3}R} V_0 = \frac{2}{5} V_0$

If $V_0 = 0$ Then $V_- = \beta V_2$



$V_- = \frac{\frac{3}{2}R}{R + \frac{3}{2}R} V_2 = \frac{3}{5} V_2$

Part c. Combine the results of parts a and b to determine an expression for V_o in terms of V_1 and V_2.

V_o = $5V_1 - 3V_2$

$V_0 = \frac{V_- - \beta}{\alpha} = \frac{V_1 - \frac{3}{5}V_2}{\frac{1}{5}} = 3V_1 - 3V_2$

3 OOP (14 points)

We wish to implement a set of classes to model graphs of the type that we used to describe search problems.

Graphs have **nodes** and **edges**. A directed edge indicates a connection from the first node to the second. An undirected edge can be thought of as two directed edges, one from the first node to the second and another one going the other way. Nodes will be identified with names that are represented using Python strings.

Implement a base class called `DirectedGraph`, which has the following methods:

- `hasNode` - given a node, returns `True` when the node exists in the graph and false otherwise.
- `addNode` - given a node, if it's not already present, it adds it to the graph, initially with no edges.
- `addEdge` - given two nodes, add a connection from the first to the second. If either node is not already present, it is added to the graph.
- `children` - given a node, returns a list of nodes that can be reached from that node by traversing a single arc. If the node is not present, it returns an empty list.

Here is an example use:

```
>>> g = DirectedGraph()
>>> g.addNode('A')
>>> g.addEdge('A','B')
>>> g.children('A')
['B']
>>> g.children('B')
[]
```

Think very carefully about:

- How you will keep track of the nodes in the graph.
- How you will keep track of which nodes are connected.

Each of the methods should be short.

Define this class and its methods.

```
class DirectedGraph:
    def __init__(self):
        self.data = {}  <- can just have and will be there
    def hasNode(self, node):
        if self.data[node]:
            return True
        else:
            return False
    def addNode(self, node):  <- check if .. not self.hasNode(node)
        self.data[node] = []
    def addEdge(self, start, end):
        if not self.hasNode(start):
            self.addNode(start)  <- or just call - add should check
        #some w end
        self.data[start].append(end)
    def children(node):
        if hasNode[node]:
            return self.data[node]
        else:
            return []
```

Define a class `UnDirectedGraph` which has all the same methods as `DirectedGraph`, except that `addEdge`, adds edges in both directions. Here is an example use:

```
>>> g = UnDirectedGraph()
>>> g.addEdge('A','B')
>>> g.children('A')
['B']
>>> g.children('B')
['A']
```

Define this class. Use inheritance to avoid rewriting all of the methods.

```
class UnDirectedGraph(DirectedGraph):
    def addEdge(self, start, end):
        # repeat the add check
        # Same add first
        DirectedGraph.addEdge(start, end)
        self.data[end].append(start)
                    # or call again w/ (end, start)
```

4 Graph Search (14 points)

We can use the `DirectedGraph` class (previous problem) to define a search problem. We will define a state machine class, `GraphSM`, whose states correspond to the nodes in the graph and whose state transitions correspond to the edges in the graph.

The input to the machine is an integer indicating which of the children of the current node will become the next state. If the input is greater than or equal to the length of the list of children, then the state remains unchanged.

A `GraphSM` machine is initialized with an instance of `DirectedGraph`, a start node and an integer indicating the maximum number of children of any node in the graph.

```
>>> g = DirectedGraph()
>>> for s in ['S', 'A', 'B', 'C', 'D', 'E', 'F', 'G']:
        g.addNode(s)
>>> for (a, b) in [('S', 'A'), ('S', 'B'), ('A', 'C'), ('B', 'C'), ('B', 'D'),
                   ('C', 'E'), ('D', 'E'), ('D', 'F'), ('F', 'G')]:
        g.addEdge(a,b)
>>> m = GraphSM(g, 'S', 2)
>>> m.transduce([1,1,1,0])
['B', 'D', 'F', 'G']
>>> search.smSearch(GraphSM(g, 'S', 2), goalTest=lambda s: s == 'E')
[(None, 'S'), (0, 'A'), (0, 'C'), (0, 'E')]
```

4.1 GraphSM

Define the `GraphSM` class. Define all the methods and attributes that you need for the state machine to be used for a search, as shown in the example above. You do not need to define a done method.

```
class GraphSM(SM):
    def __init__(self, graph, startNode, maxChildren):
        self.graph = graph
        self.maxChildren = maxChildren
        self.startState = startNode
    def getNextValue(self, state, inp):
        newState = self.graph.children(state)[inp]
        return (newState, newState)
```

they make legal inputs = range(max kids)

sm built in

Oh need handeling for max children

try — don't remember

catch

if inp > maxChildren:

if self. . . .

else:
    newState = state

if inp < len(c)
    state = c[inp]

they don't use max children!

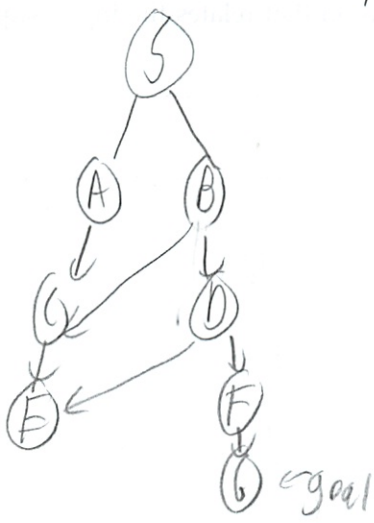
legal inputs might be part of SM
—for it to search over — it is — but wh, do you need?

4.2 Search

For the graph defined by the code shown above:

- Show the sequence of partial paths **expanded** by depth-first search **without** DP from S to G. **Assume children of a state are pushed on the agenda (visited) in reverse alphabetical order.** There are more than enough slots.

step	partial path expanded
1	S
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

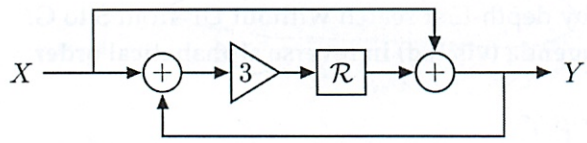


- List in order the states that are **visited** by breadth-first search **without** DP starting at S and going to G. If a state is visited more than once, include it each time it is visited.

- List in order the states that are **visited** by breadth-first search **with** DP starting at S and going to G. If a state is visited more than once, include it each time it is visited.

5 System Functions (15 points)

Consider the following system.



diff eq = $\frac{Y}{X}$

Part a. Enter the difference equation that relates the input sequence x[n] and output sequence y[n] for this system.

$y[n] = x[n] + 3x[n-1] + 3y[n-1]$ ✓

$Y = X + 3XR + 3RY$

$Y(1-3R) = X(1+3R)$

$\frac{Y}{X} = \frac{1+3R}{1-3R}$

Part b. Enter the pole(s) of this system in the box below. If there are multiple poles, separate them with commas. If there are no poles, enter none.

Ca. I do this?

3

$1-3R = 0$

$1 = 3R$

$R = \frac{1}{3}$

Must convert to z

$z - 3$

$z = 3$

replace R w/ $\frac{1}{z}$

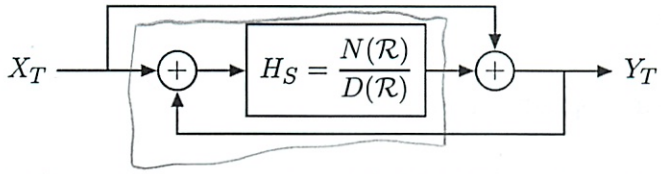
then * by z^2 ← keep forgetting

← here multiply by z

don't be thrown off by ease!

$\frac{z+3}{z-3}$

Part c. Consider a generalization of the previous system with the following form.



Write a Python function called `YinYang` that takes a single input parameter `HS`, which is a `SystemFunction` that represents H_S, and returns a `SystemFunction` that represents $H_T = Y_T(\mathcal{R})/X_T(\mathcal{R})$.

```
def YinYang(HS):
    in = sf.Gain(1)
    out = sf.Gain(1)
    first = sf.FeedforwardAdd(in, HS)
    return sf.FeedbackAdd(first)
```

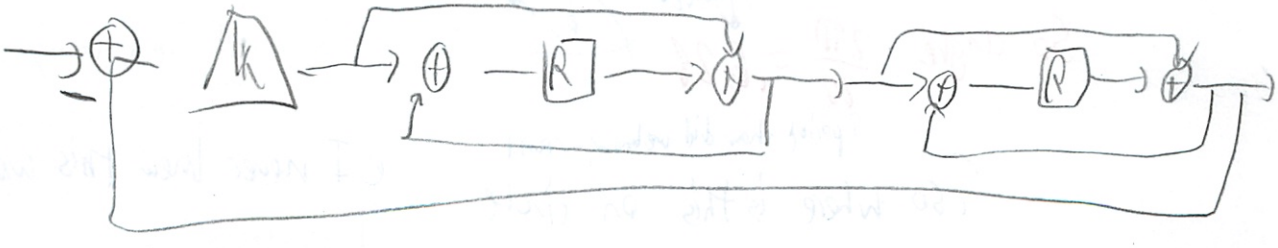
```
return sf.SystemFunction(HS.denominator + HS.numerator,
                         HS.denominator - HS.numerator)
```

Part d. Consider the following code, which builds on the `YinYang` function in part c.

```
def TwoYinYangs(K):
    g1 = sf.Gain(K)
    h1 = YinYang(sf.R())
    h2 = YinYang(sf.R())
    return sf.FeedbackSubtract(sf.Cascade(g1,sf.Cascade(h1,h2)))
```

Enter the system function that will be returned by `TwoYinYangs(3)` in the box below. Express your answer as a ratio of polynomials in $\mathcal{R}$.

$H(\mathcal{R}) =$



Part e. Executing the following code

diff gains

```
for K in [0.01, 0.02, 0.05, 0.1, 0.2, 0.5]:
    print K
    for p in TwoYinYangs(K).poles():
        print ">> x,y= ",p,"    r,theta= ",sf.complexPolar(p)
```

produces the following output.

```
0.01
>> x,y=  (0.980+0.198j)     r,theta=  (1.0,  0.199)
>> x,y=  (0.980-0.198j)     r,theta=  (1.0, -0.199)
0.02
>> x,y=  (0.961+0.277j)     r,theta=  (1.0,  0.281)
>> x,y=  (0.961-0.277j)     r,theta=  (1.0, -0.281)
0.05
>> x,y=  (0.905+0.426j)     r,theta=  (1.0,  0.440)
>> x,y=  (0.905-0.426j)     r,theta=  (1.0, -0.440)
0.1
>> x,y=  (0.818+0.575j)     r,theta=  (1.0,  0.613)
>> x,y=  (0.818-0.575j)     r,theta=  (1.0, -0.613)
0.2
>> x,y=  (0.667+0.745j)     r,theta=  (1.0,  0.841)
>> x,y=  (0.667-0.745j)     r,theta=  (1.0, -0.841)
0.5
>> x,y=  (0.333+0.943j)     r,theta=  (1.0 , 1.231)
>> x,y=  (0.333-0.943j)     r,theta=  (1.0, -1.231)
```

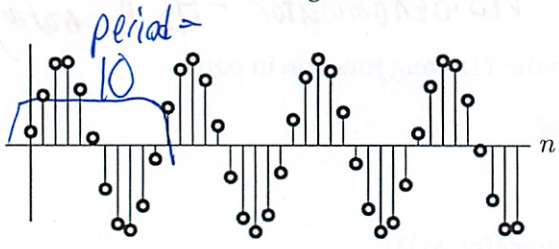
all on circle

$\frac{2\pi}{\text{period}}$ = phase

look/count till it repeats

remember what period is

Which value of K would give rise to the following unit sample response?



a little decay

K =? (0.01 or 0.02 or 0.05 or 0.1 or 0.2 or 0.5 or **none**) : none !!

mag dom pole = 1

period ≈ 10

So angle $\frac{2\pi}{10} = .628$

phase/freq = angle

outside circle

here's

no real, all imaginary

period - how did we know - can't

so where is this on circle

closest to its peak

I never knew this worked

Cascade $= H_2 H_1$

$$\frac{1+R}{1-R}$$

Top

$$3 \circ \frac{1+R}{1-R} \circ \frac{1+R}{1-R}$$

Blacks top $\mathcal{T} H_1 \quad H_2 = 1$

$$\frac{H_1}{1+H_1+H_2}$$

$$\frac{3 \circ \frac{1+R}{1-R} \circ \frac{1+R}{1-R}}{1 + 1 \circ 3 \circ \frac{1+R}{1-R} \circ \frac{1+R}{1-R}}$$

$$3 \circ \frac{1+R}{1-R} \circ \frac{1+R}{1-R}$$

$$3 \cdot \frac{1+2R+R^2}{1-2R+R^2}$$

$$\frac{3+6R+3R^2}{1-2R+R^2}$$

$$\frac{\frac{3+6R+3R^2}{1-2R+R^2}}{1+\frac{3+6R+3R^2}{1-2R+R^2}} \cdot \frac{1-2R+R^2}{1-2R+R^2}$$

$$\frac{3+6R+3R^2}{1-2R+R^2+3+6R+3R^2}$$

$$\frac{3+6R+3R^2}{4R^2+4R+4}$$

correct

$$\frac{3(1+R)^2}{(1-R)^2+3(1+R)^2} = \frac{3}{4}\,\frac{1+2R+R^2}{1+R+R^2}$$

I think I was close

– or right – but unsimplified

↑ I was correct

just not simplified

6 State Estimation (14 points)

We are interested in estimating and predicting the amount of traffic on a particular segment of road. We will model the traffic level as being either Low, Medium, or High.

Observations

When a specially-equipped car drives down the road of interest, we can measure its speed. Knowing the car's speed gives us information about the traffic on the road. The **conditional probability distribution** Pr(*Speed* | *Traffic*) is specified in the following table.

	Speed							
Traffic	<10	10-20	20-30	30-40	40-50	50-60	60-70	70-80
Low	0	0.1	0.1	0.1	0.2	0.2	0.2	0.1
Med	0.2	0.2	0.2	0.2	0.2	0	0	0
High	0.5	0.5	0	0	0	0	0	0

Part a. Imagine that our initial belief about the traffic is:

$$\Pr(S_0 = Low) = 0.5, \quad \Pr(S_0 = Med) = 0.25, \quad \Pr(S_0 = High) = 0.25.$$

Now, we make a single observation O_0 of the speed of a car on the road.
In the following, we consider three possible outcomes: $O_0 = 75$ or $O_0 = 45$ or $O_0 = 5$.

- What would be our updated belief given that the single observation is $O_0 = 75$?

$\Pr(S_0 = Low \mid O_0 = 75) =$ 1 $\Pr(S_0 = Med \mid O_0 = 75) =$ 0

$\Pr(S_0 = High \mid O_0 = 75) =$ 0

- What would be our updated belief given that the single observation is $O_0 = 45$?

$\Pr(S_0 = Low \mid O_0 = 45) =$ 2/3 $\Pr(S_0 = Med \mid O_0 = 45) =$ 1/3

$\Pr(S_0 = High \mid O_0 = 45) =$ 0

- What would be our updated belief given that the single observation is $O_0 = 5$?

$\Pr(S_0 = Low \mid O_0 = 5) =$ 0 $\Pr(S_0 = Med \mid O_0 = 5) =$ 2/7

$\Pr(S_0 = High \mid O_0 = 5) =$ 5/7

Transitions

Now, consider how the traffic state changes over time. We'll think about the traffic system as having two possible inputs (actions), N, and A. Input I_t = N means that there has been no disturbance on that stretch of road on step t; input I_t = A means that there has been an accident on that stretch of road at step t. Depending on whether there has or has not been an accident, the transition probabilities differ.

The **conditional probability distribution** $\Pr(S_{t+1} \mid S_t, I_t = N)$ is given by:

		S_t+1		
		Low	Med	High
	Low	0.9	0.1	0
S_t	Med	0.1	0.8	0.1
	High	0	0.1	0.9

The **conditional probability distribution** $\Pr(S_{t+1} \mid S_t, I_t = A)$ is given by:

		S_t+1		
		Low	Med	High
	Low	0	0.1	0.9
S_t	Med	0	0	1
	High	0	0	1

Part b1. If we are certain that the traffic is low at time 0 (that is, $\Pr(S_0 = \mathit{Low}) = 1$), and there are no accidents for the next two steps (and we make no observations), what is our belief about the state of the traffic at time 2? Specify the following values:

$\Pr(S_2 = \mathit{Low} \mid I_0 = N, I_1 = N) =$ 1 DDist{.9, .1, 0} 2: DDist{.81+.01, .08+.09, .01}

$\Pr(S_2 = \mathit{Med} \mid I_0 = N, I_1 = N) =$

$\Pr(S_2 = \mathit{High} \mid I_0 = N, I_1 = N) =$

Part b2. If we are certain that the traffic is low at time 0, and there are accidents for the next two steps (and we make no observations), what is our belief about the state of the traffic at time 2? Specify the following values:

$\Pr(S_2 = \mathit{Low} \mid I_0 = A, I_1 = A) =$ 1: 0 | 0·0 + 0·0 + 0·0 = 0

$\Pr(S_2 = \mathit{Med} \mid I_0 = A, I_1 = A) =$ 1: .1 | .1·.1 + 0·0 + 0·0 = .01

$\Pr(S_2 = \mathit{High} \mid I_0 = A, I_1 = A) =$ 1: .9 | .9·0 + .1·.1 + .9·1 = .01 + .9 1

Combining observation and transition

Part c. Assuming that you initially believe that each traffic state is equally likely, which of the following sequences are possible?

$O_0=75, I_0=A, O_1=75, I_1=N$

possible? (yes/no): No If no, briefly explain why (one sentence only).

If I_0 accident than can't see 75 speed since is

$O_0=75, I_0=A, O_1=15, I_1=N$

possible? (yes/no): Yes If no, briefly explain why (one sentence only).

Oh yes it can - but can't transition see which

$O_0=75, I_0=N, O_1=15, I_1=N$

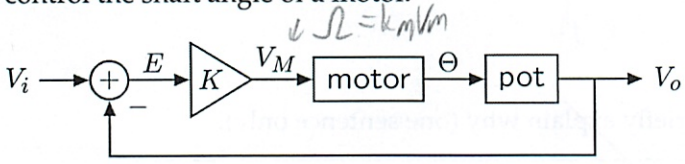
possible? (yes/no): Yes If no, briefly explain why (one sentence only).

Improbabubable but likly

lot of work for 15 pts

7 Modeling (15 points)

The following block diagram illustrates the signal flow paths through a circuit that is used to control the shaft angle of a motor.



$\Omega = k_m V_m$

? $k = k_m$

The error E between the input voltage V_i and output voltage V_o is multiplied by the gain K to generate the motor input voltage V_M. Assume that the angular speed of the motor shaft (Ω, not shown) is proportional to V_M so that $\Omega = K_M V_M$. The motor turns the shaft of a potentiometer ("pot") to angle Θ and the output voltage from the potentiometer (V_o) is proportional to the shaft angle, i.e., $V_o = \alpha\Theta$.

oh like a real problem

We wish to make a model of this system in which the shaft angle Θ at time n is equal to the shaft angle at time $(n-1)$ plus the product of the time between steps, T, times the angular speed Ω at time $(n-1)$,

assume I

? How are V_m and Ω related?

$V_m = \Omega$ — oh above $\Omega = k_m V_m$

$$\theta[n] = \theta[n-1] + T\omega[n-1]$$

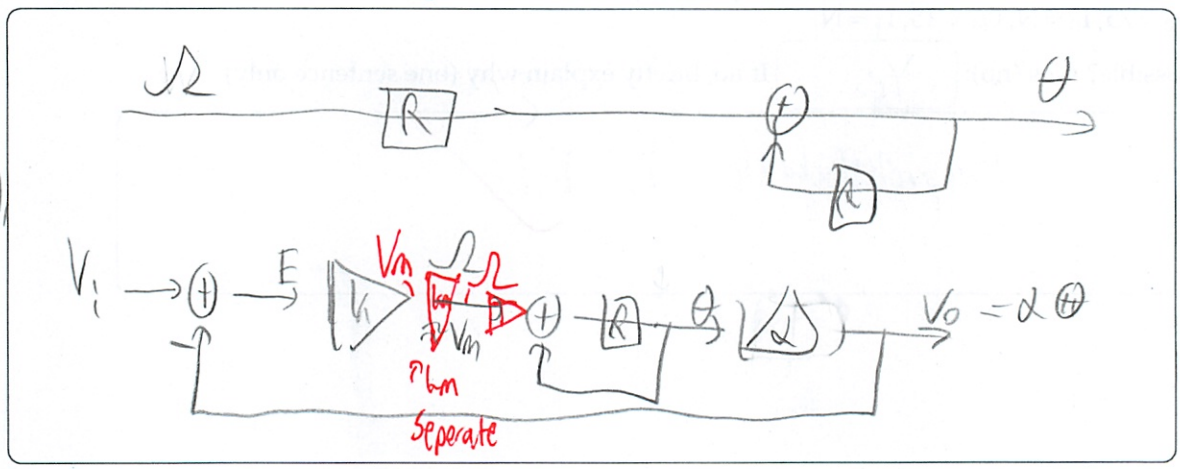
$\Theta = \theta R + \Omega R$

where $\theta[n]$ is the n^{th} sample of the Θ signal and $\omega[n]$ is the n^{th} sample of the Ω signal.

$Y(1-R) = \Omega R$

$\frac{\Theta}{\Omega} = \frac{R}{1-R}$

Part a. Draw a block diagram for the **entire** motor system that consists of just adders, gains, and delays. Label the nodes that correspond to V_i, V_o, E, V_M, Ω, and Θ.



Cascade $\frac{kR\alpha}{1-R} = H$

For blocks $\frac{kR\alpha}{1-R}$

$\frac{\frac{kR\alpha}{1-R}}{1 + 1 \cdot \frac{kR\alpha}{1-R}}$

$\frac{kR\alpha}{1-R+kR\alpha}$

$\frac{kR\alpha}{k\alpha R - R + 1}$

Part b. Determine the system function $H = \frac{V_o}{V_i}$. Express your answer as a ratio of two polynomials in $\mathcal{R}$.

just more ks

$$H = \frac{V_o}{V_i} = \frac{k\alpha R}{k\alpha R - R + 1} \qquad \frac{k k_m T \alpha R}{1 - R + k k_m T \alpha R}$$

↓ need to do cascade

- just read the problem better I guess

$V_o = \alpha k V_i[n-1] + T\omega[n-1]$

Casca

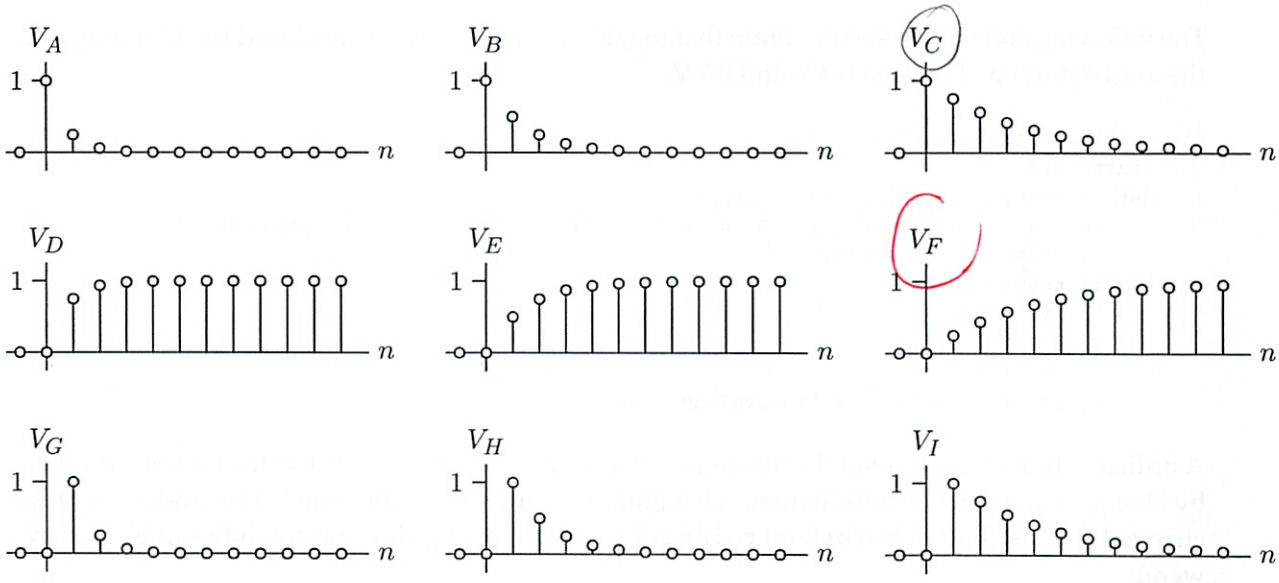
start / steady state

Part c. The step response of the system was measured by stepping the input voltage from 0 V for times $n < 0$ to 1 V for times $n \geq 0$. Based on this measurement, it was concluded that the system function can be written in the following form:

$$H = \frac{V_o}{V_i} = \frac{\frac{1}{4}\mathcal{R}}{1 - \frac{3}{4}\mathcal{R}}$$

filling in constants

Which of the following **step responses** is consistent with the previous equation?



step response = V_A or V_B or ... or V_I or none: C F

like earlier qu

$1 - \frac{3}{4}z \cdot z$?

$z - \frac{3}{4} = 0$

$z = \frac{3}{4}$

look at start

run a few times and see what happens

really good suggestion since I am bad at reading circle

bounded

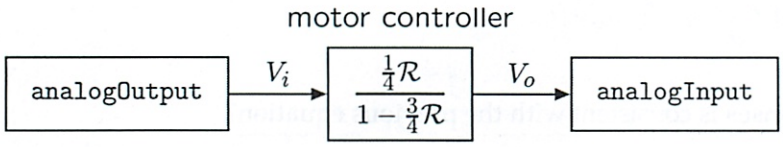
increase or decrease monotonically

Why?

- still converges to a point

How to know that?

Part d. The motor control circuit was wired up to the robot so that the robot's "brain" could "command" the motor input voltage V_i from the `analogOutput` and "read" the resulting motor output voltage V_o via an `analogInput`.



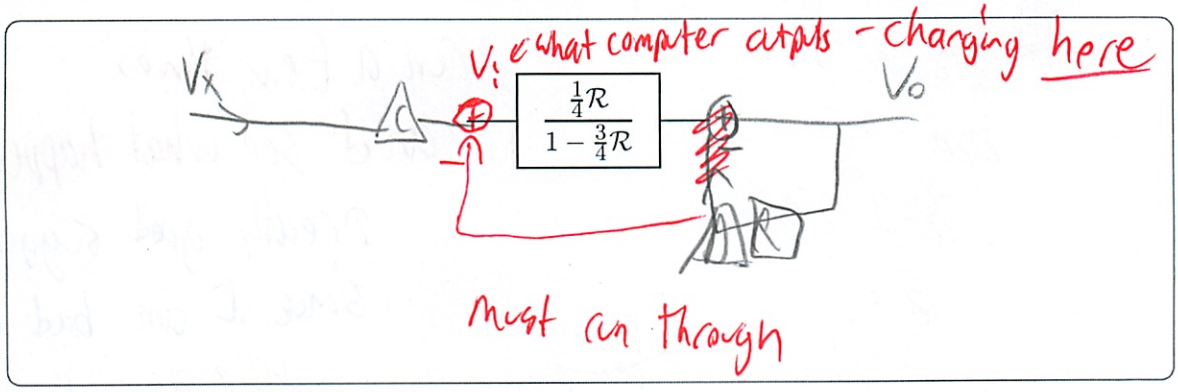
The following code implements a brain that toggles the motor position back and forth by toggling the `analogOutput` between 0.3 V and 0.7 V.

```
class motorControlBrain(sm.SM):
    startState = 0
    def getNextValues(self, state, inp):
        vo = inp.analogInput[0]   # vo is the output voltage from the previous step
        newState = (state+1)%100
        if newState>50:
            vx = 0.7
        else:
            vx = 0.3
        return (newState,io.Action(voltage=vx))
```

A brilliant student figured out that it was possible to speed up the response of the motor controller by changing the control **code** without changing the hardware in any way! The student simply changed the last line of `motorControlBrain` to the following (leaving the other lines as they were):

```
        return (newState,io.Action(voltage=C*vx-D*vo))
```

where `C` and `D` are constants. We will refer to the resulting system as the "CD Motor Controller." Draw a block diagram that relates the input V_X to the output V_o for the CD Motor Controller. Your block diagram should include the original motor controller, which is already provided below.



Part e. Determine the system function $H_X = \frac{V_o}{V_X}$ for the CD Motor Controller.

$H_X = \frac{V_o}{V_X} =$ $\frac{C\frac{1}{4}R}{-\frac{3}{4}DR^2+RD-\frac{3}{4}R+1}$ $\frac{\frac{1}{4}CR}{1-\frac{3}{4}R+\frac{1}{4}DR^2}$

Cascade

Now part $V_o = V_x[n] - V_o[n-1]D$

$Y = X - YRD$

$Y(1+RD) = X$

$\frac{Y}{X} = \frac{1}{1+RD}$

I drew pic wrong - simpler

$\frac{C}{1} \circ \frac{\frac{1}{4}R}{1-\frac{3}{4}R} \cdot \frac{1}{1+RD} = \frac{C\frac{1}{4}R}{(1-\frac{3}{4}R+RD-\frac{3}{4}DR^2)}$

Part f. To use the CD Motor Controller, our brilliant student had to determine values of the constants C and D. The values that were chosen gave rise to the following system function:

$$H_X = \frac{V_o}{V_X} = \frac{\frac{25}{64}\mathcal{R}}{1-\frac{3}{4}\mathcal{R}+\frac{9}{64}\mathcal{R}^2}.$$

Can use these to check

Is the performance of this system better or worse than that of the original motor controller in part c? Briefly explain your reasoning.

1. find roots

$a = 1$ $b = -\frac{3}{4}$ $c = \frac{9}{64}$

$\frac{-b \pm \sqrt{b^2-4ac}}{2a}$

$\frac{\frac{3}{4}+\sqrt{0}}{2\cdot 1}$ ✓

$= \frac{3}{8}$

~~lower gain so slightly worse~~

Smaller mag of dom pole — converges faster

Oh duh I remeber that — I shald really memorize a char

No Circuits!

6.01 Final Exam: Spring 2009

Name:

Enter all answers in the boxes provided.
You may use any paper materials you have brought.
No computers, cell phones, or music players.

For staff use:

1.	/14
2.	/14
3.	/14
4.	/14
5.	/15
6.	/14
7.	/15
total:	/100

worksheet

1 Programming (14 points)

In the Equation class, we represent a linear equation

$$a_0x_0 + ... + a_nx_n = c$$

with a list of coefficients, `coeffs`, corresponding to $[a_0, ..., a_n]$ and the `constant`, `c`.

```
class Equation:
    def __init__(self, coeffs, variableNames, constant):
        self.variableNames = variableNames    # List of variable names
        self.coeffs = coeffs                  # List of coefficients
        self.constant = constant              # Constant (right hand side)
```

Suppose that we had a list of values of `[v0, ..., vn]` for the variables $(x_0, ... x_n)$ in which one of the values is `None` and all of the rest of the values are numbers.

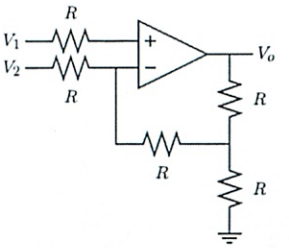
These numeric values are enough to determine a value for the unspecified variable.

Write a procedure `determineValue` that takes an instance of the `Equation` class and a list of values (exactly one of which is `None`) and returns the value of the unspecified variable.

```
def determineValue(eq, values):
    ans = eq.constant
    for i in range(len(eq.coeffs)):
        if values[i] == None:
            a = eq.coeffs[i]
        else:
            ans -= eq.coeffs[i]*values[i]
    return ans/float(a)
```


2 Circuits (14 points)

Let V_- represent the voltage at the negative input of the op-amp in the following circuit.



Notice that if we knew the values of V_o and V_2 then we could calculate the value of V_-. The resulting value of V_- will be a linear combination of V_2 and V_o of the following form:

$$V_- = \alpha V_o + \beta V_2. \quad (1)$$

Part a. Use Equation 1 to find an expression for the output voltage V_o in terms of the input voltages V_1 and V_2. Which of the following expressions gives the result?

Hint: it is not necessary to determine the values of α and β to do this part of the question.

$$V_A = \alpha V_2 - \beta V_1 \qquad V_B = \frac{\alpha V_2 - V_1}{\beta} \qquad V_C = \beta V_2 - \alpha V_1$$

$$V_D = \frac{\beta V_2 - V_1}{\alpha} \qquad V_E = \frac{V_2 - \alpha V_1}{\beta} \qquad V_F = \frac{V_2 - \beta V_1}{\alpha}$$

$$V_G = \alpha V_1 - \beta V_2 \qquad V_H = \frac{\alpha V_1 - V_2}{\beta} \qquad V_I = \beta V_1 - \alpha V_2$$

$$V_J = \frac{\beta V_1 - V_2}{\alpha} \qquad V_K = \frac{V_1 - \alpha V_2}{\beta} \qquad V_L = \frac{V_1 - \beta V_2}{\alpha}$$

V_o = (V_A or V_B or ... or V_L or none) V_L

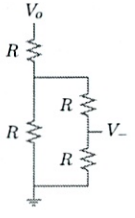
$V_1 = V_+ = V_- = \alpha V_o + \beta V_2$

$V_o = \frac{V_1 - \beta V_2}{\alpha}$

Part b. Find α and β in Equation 1 (previous page). **Hint:** Try superposition.

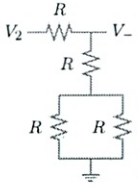
$\alpha =$ $\frac{1}{5}$ $\beta =$ $\frac{3}{5}$

Notice that if $V_2 = 0$ then $V_- = \alpha V_o$.



$$V_- = \frac{1}{2}V_x \qquad V_x = \frac{2R\|R}{R + 2R\|R}V_o = \frac{\frac{2}{3}R}{R + \frac{2}{3}R}V_o = \frac{2}{5}V_o$$

Also, if $V_o = 0$ then $V_- = \beta V_2$.



$$V_- = \frac{\frac{3}{2}R}{R + \frac{3}{2}R}V_2 = \frac{3}{5}V_2$$

Part c. Combine the results of parts a and b to determine an expression for V_o in terms of V_1 and V_2.

$V_o =$ $5V_1 - 3V_2$

$$V_o = \frac{V_1 - \beta V_2}{\alpha} = \frac{V_1 - \frac{3}{5}V_2}{\frac{1}{5}} = 5V_1 - 3V_2$$

Grading: 4 points for part a; 4 points for α; 4 points for β; 2 points for part c.

3 OOP (14 points)

We wish to implement a set of classes to model graphs of the type that we used to describe search problems.

Graphs have **nodes** and **edges**. A directed edge indicates a connection from the first node to the second. An undirected edge can be thought of as two directed edges, one from the first node to the second and another one going the other way. Nodes will be identified with names that are represented using Python strings.

Implement a base class called `DirectedGraph`, which has the following methods:

- `hasNode` - given a node, returns `True` when the node exists in the graph and false otherwise.
- `addNode` - given a node, if it's not already present, it adds it to the graph, initially with no edges.
- `addEdge` - given two nodes, add a connection from the first to the second. If either node is not already present, it is added to the graph.
- `children` - given a node, returns a list of nodes that can be reached from that node by traversing a single arc. If the node is not present, it returns an empty list.

Here is an example use:

```
>>> g = DirectedGraph()
>>> g.addNode('A')
>>> g.addEdge('A','B')
>>> g.children('A')
['B']
>>> g.children('B')
[]
```

Think very carefully about:

- How you will keep track of the nodes in the graph.
- How you will keep track of which nodes are connected.

Each of the methods should be short.

Define this class and its methods.

```
class DirectedGraph:
    nodes = {}
    def hasNode(self,n):
        return n in self.nodes
    def addNode(self,n):
        if not self.hasNode(n):
            self.nodes[n] = {}
    def addEdge(self,n,c):
        self.addNode(n)
        self.addNode(c)
        if not (c in self.nodes[n]):
            self.nodes[n][c] = True
    def children(self,n):
        return self.nodes[n].keys()
```


Define a class `UnDirectedGraph` which has all the same methods as `DirectedGraph`, except that `addEdge`, adds edges in both directions. Here is an example use:

```
>>> g = UnDirectedGraph()
>>> g.addEdge('A','B')
>>> g.children('A')
['B']
>>> g.children('B')
['A']
```

Define this class. Use inheritance to avoid rewriting all of the methods.

```
class UnDirectedGraph(DirectedGraph):
    def addEdge(self,n,c):
        DirectedGraph.addEdge(self,n,c)
        DirectedGraph.addEdge(self,c,n)
```

4 Graph Search (14 points)

We can use the `DirectedGraph` class (previous problem) to define a search problem. We will define a state machine class, `GraphSM`, whose states correspond to the nodes in the graph and whose state transitions correspond to the edges in the graph.

The input to the machine is an integer indicating which of the children of the current node will become the next state. If the input is greater than or equal to the length of the list of children, then the state remains unchanged.

A `GraphSM` machine is initialized with an instance of `DirectedGraph`, a start node and an integer indicating the maximum number of children of any node in the graph.

```
>>> g = DirectedGraph()
>>> for s in ['S', 'A', 'B', 'C', 'D', 'E', 'F', 'G']:
        g.addNode(s)
>>> for (a, b) in [('S', 'A'), ('S', 'B'), ('A', 'C'), ('B', 'C'), ('B', 'D'),
                   ('C', 'E'), ('D', 'E'), ('D', 'F'), ('F', 'G')]:
        g.addEdge(a,b)
>>> m = GraphSM(g, 'S', 2)
>>> m.transduce([1,1,1,0])
['B', 'D', 'F', 'G']
>>> search.smSearch(GraphSM(g, 'S', 2), goalTest=lambda s: s == 'E')
[(None, 'S'), (0, 'A'), (0, 'C'), (0, 'E')]
```


4.1 GraphSM

Define the `GraphSM` class. Define all the methods and attributes that you need for the state machine to be used for a search, as shown in the example above. You do not need to define a done method.

```
class GraphSM(SM):
    def __init__(self, graph, startNode, maxChildren):
        self.graph = graph
        self.startState = startNode
        self.MaxChildren = maxChildren
        self.legalInputs = range(maxChildren)

    def getNextValues(self, state, inp)
        c = self.graph.children(state)
        if inp<len(c):
            state = c[inp]
        return (state,state)
```

4.2 Search

For the graph defined by the code shown above:

- Show the sequence of partial paths **expanded** by depth-first search **without** DP from S to G. **Assume children of a state are pushed on the agenda (visited) in reverse alphabetical order.** There are more than enough slots.

step	partial path expanded
1	S
2	SA
3	SAC
4	SACE
5	SB
6	SBC
7	SBCE
8	SBD
9	SBDE
10	SBDF
11	
12	

The final path is SBDFG.

- List in order the states that are **visited** by breadth-first search **without** DP starting at S and going to G. If a state is visited more than once, include it each time it is visited.

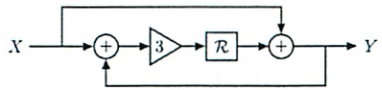
SBADCCFEEEG

- List in order the states that are **visited** by breadth-first search **with** DP starting at S and going to G. If a state is visited more than once, include it each time it is visited.

SBADCFEG

5 System Functions (15 points)

Consider the following system.



Part a. Enter the difference equation that relates the input sequence x[n] and output sequence y[n] for this system.

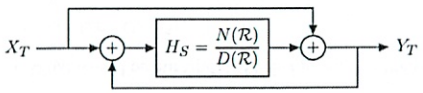
$y[n] = x[n] + 3x[n-1] + 3y[n-1]$

Part b. Enter the pole(s) of this system in the box below. If there are multiple poles, separate them with commas. If there are no poles, enter none.

3

$$H = \frac{Y}{X} = \frac{1+3\mathcal{R}}{1-3\mathcal{R}} = \frac{z+3}{z-3}$$

Part c. Consider a generalization of the previous system with the following form.



Write a Python function called `YinYang` that takes a single input parameter `HS`, which is a `SystemFunction` that represents H_S, and returns a `SystemFunction` that represents $H_T = Y_T(\mathcal{R})/X_T(\mathcal{R})$.

```
def YinYang(HS):
    return sf.SystemFunction(HS.denominator+HS.numerator,HS.denominator-HS.numerator)
```

Part d. Consider the following code, which builds on the `YinYang` function in part c.

```
def TwoYinYangs(K):
    g1 = sf.Gain(K)
    h1 = YinYang(sf.R())
    h2 = YinYang(sf.R())
    return sf.FeedbackSubtract(sf.Cascade(g1,sf.Cascade(h1,h2)))
```

Enter the system function that will be returned by `TwoYinYangs(3)` in the box below. Express your answer as a ratio of polynomials in $\mathcal{R}$.

$$H(\mathcal{R}) = \frac{3(1+\mathcal{R})^2}{(1-\mathcal{R})^2 + 3(1+\mathcal{R})^2} = \frac{3}{4} \setminus \frac{1+2\mathcal{R}+\mathcal{R}^2}{1+\mathcal{R}+\mathcal{R}^2}$$

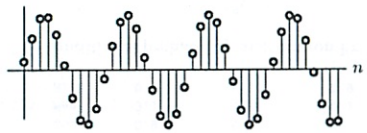
Part e. Executing the following code

```
for K in [0.01, 0.02, 0.05, 0.1, 0.2, 0.5]:
    print K
    for p in TwoYinYangs(K).poles():
        print ">> x,y= ",p,"    r,theta= ",sf.complexPolar(p)
```

produces the following output.

```
0.01
>> x,y=  (0.980+0.198j)     r,theta=  (1.0,  0.199)
>> x,y=  (0.980-0.198j)     r,theta=  (1.0, -0.199)
0.02
>> x,y=  (0.961+0.277j)     r,theta=  (1.0,  0.281)
>> x,y=  (0.961-0.277j)     r,theta=  (1.0, -0.281)
0.05
>> x,y=  (0.905+0.426j)     r,theta=  (1.0,  0.440)
>> x,y=  (0.905-0.426j)     r,theta=  (1.0, -0.440)
0.1
>> x,y=  (0.818+0.575j)     r,theta=  (1.0,  0.613)
>> x,y=  (0.818-0.575j)     r,theta=  (1.0, -0.613)
0.2
>> x,y=  (0.667+0.745j)     r,theta=  (1.0,  0.841)
>> x,y=  (0.667-0.745j)     r,theta=  (1.0, -0.841)
0.5
>> x,y=  (0.333+0.943j)     r,theta=  (1.0 , 1.231)
>> x,y=  (0.333-0.943j)     r,theta=  (1.0, -1.231)
```

Which value of K would give rise to the following unit sample response?



K =? (0.01 or 0.02 or 0.05 or 0.1 or 0.2 or 0.5 or **none**) : 0.1

The unit sample response oscillates with little decay. Therefore, the magnitude of the dominant pole must be 1. The period is approximately 10. Therefore, the angle of the dominant pole must be approximately $2\pi/10 \approx 0.628$, which is close to the answer for K = 0.1.

Grading: 3 points for each part.

6 State Estimation (14 points)

We are interested in estimating and predicting the amount of traffic on a particular segment of road. We will model the traffic level as being either Low, Medium, or High.

Observations

When a specially-equipped car drives down the road of interest, we can measure its speed. Knowing the car's speed gives us information about the traffic on the road. The **conditional probability distribution** $\Pr(Speed \mid Traffic)$ is specified in the following table.

	Speed							
Traffic	<10	10-20	20-30	30-40	40-50	50-60	60-70	70-80
Low	0	0.1	0.1	0.1	0.2	0.2	0.2	0.1
Med	0.2	0.2	0.2	0.2	0.2	0	0	0
High	0.5	0.5	0	0	0	0	0	0

Part a. Imagine that our initial belief about the traffic is:

$$\Pr(S_0 = Low) = 0.5, \quad \Pr(S_0 = Med) = 0.25, \quad \Pr(S_0 = High) = 0.25.$$

Now, we make a single observation O_0 of the speed of a car on the road.
In the following, we consider three possible outcomes: $O_0 = 75$ or $O_0 = 45$ or $O_0 = 5$.

- What would be our updated belief given that the single observation is $O_0 = 75$?

$\Pr(S_0 = Low \mid O_0 = 75) =$ 1 $\Pr(S_0 = Med \mid O_0 = 75) =$ 0

$\Pr(S_0 = High \mid O_0 = 75) =$ 0

- What would be our updated belief given that the single observation is $O_0 = 45$?

$\Pr(S_0 = Low \mid O_0 = 45) =$ 2/3 $\Pr(S_0 = Med \mid O_0 = 45) =$ 1/3

$\Pr(S_0 = High \mid O_0 = 45) =$ 0

- What would be our updated belief given that the single observation is $O_0 = 5$?

$\Pr(S_0 = Low \mid O_0 = 5) =$ 0 $\Pr(S_0 = Med \mid O_0 = 5) =$ 2/7

$\Pr(S_0 = High \mid O_0 = 5) =$ 5/7

Transitions

Now, consider how the traffic state changes over time. We'll think about the traffic system as having two possible inputs (actions), N, and A. Input I_t = N means that there has been no disturbance on that stretch of road on step t; input I_t = A means that there has been an accident on that stretch of road at step t. Depending on whether there has or has not been an accident, the transition probabilities differ.

The **conditional probability distribution** $\Pr(S_{t+1} \mid S_t, I_t = N)$ is given by:

		S_t+1		
	Low	Med	High
	Low	0.9	0.1	0
S_t	Med	0.1	0.8	0.1
	High	0	0.1	0.9

The **conditional probability distribution** $\Pr(S_{t+1} \mid S_t, I_t = A)$ is given by:

		S_t+1		
	Low	Med	High
	Low	0	0.1	0.9
S_t	Med	0	0	1
	High	0	0	1

Part b1. If we are certain that the traffic is low at time 0 (that is, $\Pr(S_0 = Low) = 1$), and there are no accidents for the next two steps (and we make no observations), what is our belief about the state of the traffic at time 2? Specify the following values:

$\Pr(S_2 = Low \mid I_0 = N, I_1 = N) =$ $0.9 \times 0.9 + 0.1 \times 0.1 = 0.82$

$\Pr(S_2 = Med \mid I_0 = N, I_1 = N) =$ $0.9 \quad 0.1 + 0.1 \times 0.8 = 0.17$

$\Pr(S_2 = High \mid I_0 = N, I_1 = N) =$ $0.1 \times 0.1 = 0.01$

Part b2. If we are certain that the traffic is low at time 0, and there are accidents for the next two steps (and we make no observations), what is our belief about the state of the traffic at time 2? Specify the following values:

$\Pr(S_2 = Low \mid I_0 = A, I_1 = A) =$ 0

$\Pr(S_2 = Med \mid I_0 = A, I_1 = A) =$ 0

$\Pr(S_2 = High \mid I_0 = A, I_1 = A) =$ 1

Combining observation and transition

Part c. Assuming that you initially believe that each traffic state is equally likely, which of the following sequences are possible?

$O_0 = 75, I_0 = A, O_1 = 75, I_1 = N$

possible? (yes/no): no If no, briefly explain why (one sentence only).

If we see an accident in step 0 ($I_0 = A$) then we cannot possibly observe a speed of 75 in step 1 ($O_1 = 75$).

$O_0 = 75, I_0 = A, O_1 = 15, I_1 = N$

possible? (yes/no): yes If no, briefly explain why (one sentence only).

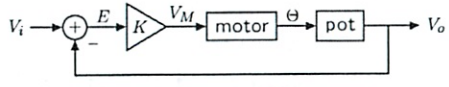
$O_0 = 75, I_0 = N, O_1 = 15, I_1 = N$

possible? (yes/no): yes If no, briefly explain why (one sentence only).

Grading: 6 points for part a; 4 points for part b; 4 points for part c

7 Modeling (15 points)

The following block diagram illustrates the signal flow paths through a circuit that is used to control the shaft angle of a motor.



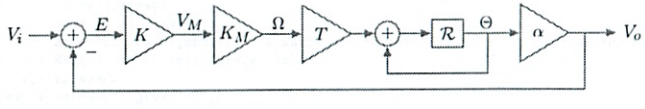
The error E between the input voltage V_i and output voltage V_o is multiplied by the gain K to generate the motor input voltage V_M. Assume that the angular speed of the motor shaft (Ω, not shown) is proportional to V_M so that $\Omega = K_M V_M$. The motor turns the shaft of a potentiometer ("pot") to angle Θ and the output voltage from the potentiometer (V_o) is proportional to the shaft angle, i.e., $V_o = \alpha\Theta$.

We wish to make a model of this system in which the shaft angle Θ at time n is equal to the shaft angle at time $(n-1)$ plus the product of the time between steps, T, times the angular speed Ω at time $(n-1)$,

$$\theta[n] = \theta[n-1] + T\omega[n-1]$$

where $\theta[n]$ is the n^{th} sample of the Θ signal and $\omega[n]$ is the n^{th} sample of the Ω signal.

Part a. Draw a block diagram for the **entire** motor system that consists of just adders, gains, and delays. Label the nodes that correspond to V_i, V_o, E, V_M, Ω, and Θ.



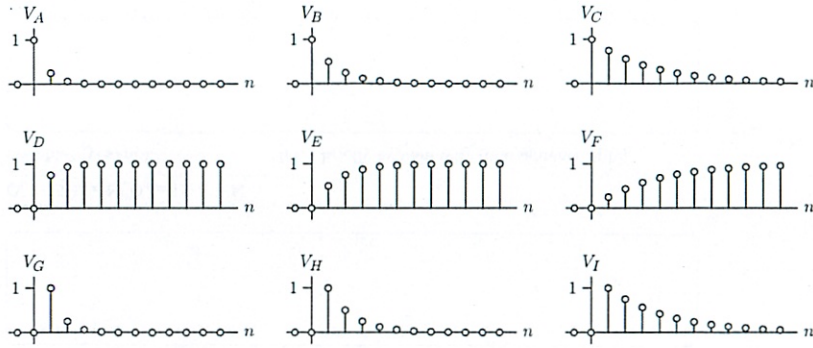
Part b. Determine the system function $H = \frac{V_o}{V_i}$. Express your answer as a ratio of two polynomials in $\mathcal{R}$.

$H = \frac{V_o}{V_i} = \frac{KK_M T\alpha\mathcal{R}}{1 - \mathcal{R} + KK_M T\alpha\mathcal{R}}$

Part c. The step response of the system was measured by stepping the input voltage from 0 V for times $n < 0$ to 1 V for times $n \geq 0$. Based on this measurement, it was concluded that the system function can be written in the following form:

$$H = \frac{V_o}{V_i} = \frac{\frac{1}{4}\mathcal{R}}{1 - \frac{3}{4}\mathcal{R}}$$

Which of the following **step responses** is consistent with the previous equation?



step response = V_A or V_B or ... or V_I or none: V_F

$$y[n] = \frac{1}{4}x[n-1] + \frac{3}{4}y[n-1]$$

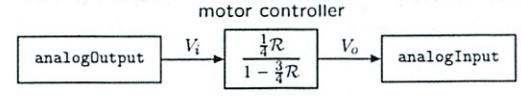
$$y[0] = \frac{1}{4}x[-1] + \frac{3}{4}y[-1] = 0$$

$$y[1] = \frac{1}{4}x[0] + \frac{3}{4}y[0] = \frac{1}{4}$$

$$y[2] = \frac{1}{4}x[1] + \frac{3}{4}y[1] = \frac{1}{4} + \frac{3}{4} \times \frac{1}{4} = \frac{7}{16}$$

$$y[3] = \frac{1}{4}x[2] + \frac{3}{4}y[2] = \frac{1}{4} + \frac{3}{4} \times \frac{7}{16} = \frac{37}{64}$$

Part d. The motor control circuit was wired up to the robot so that the robot's "brain" could "command" the motor input voltage V_i from the `analogOutput` and "read" the resulting motor output voltage V_o via an `analogInput`.



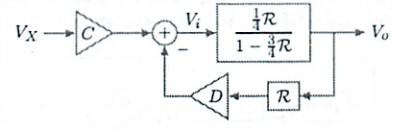
The following code implements a brain that toggles the motor position back and forth by toggling the `analogOutput` between 0.3 V and 0.7 V.

```
class motorControlBrain(sm.SM):
    startState = 0
    def getNextValues(self, state, inp):
        vo = inp.analogInput[0]   # vo is the output voltage from the previous step
        newState = (state+1)%100
        if newState>50:
            vx = 0.7
        else:
            vx = 0.3
        return (newState,io.Action(voltage=vx))
```

A brilliant student figured out that it was possible to speed up the response of the motor controller by changing the control **code** without changing the hardware in any way! The student simply changed the last line of `motorControlBrain` to the following (leaving the other lines as they were):

```
        return (newState,io.Action(voltage=C*vx-D*vo))
```

where `C` and `D` are constants. We will refer to the resulting system as the "CD Motor Controller." Draw a block diagram that relates the input V_X to the output V_o for the CD Motor Controller. Your block diagram should include the original motor controller, which is already provided below.



Part e. Determine the system function $H_X = \frac{V_o}{V_X}$ for the CD Motor Controller.

$$H_X = \frac{V_o}{V_X} = \boxed{\frac{\frac{1}{4}C\mathcal{R}}{1 - \frac{3}{4}\mathcal{R} + \frac{1}{4}D\mathcal{R}^2}}$$

$$H_X = \frac{V_o}{V_X} = C\frac{\frac{\frac{1}{4}\mathcal{R}}{1-\frac{3}{4}\mathcal{R}}}{1 + D\frac{\frac{1}{4}\mathcal{R}^2}{1-\frac{3}{4}\mathcal{R}}} = \frac{\frac{1}{4}C\mathcal{R}}{1 - \frac{3}{4}\mathcal{R} + \frac{1}{4}D\mathcal{R}^2}$$

Part f. To use the CD Motor Controller, our brilliant student had to determine values of the constants C and D. The values that were chosen gave rise to the following system function:

$$H_X = \frac{V_o}{V_X} = \frac{\frac{25}{64}\mathcal{R}}{1 - \frac{3}{4}\mathcal{R} + \frac{9}{64}\mathcal{R}^2}.$$

Is the performance of this system better or worse than that of the original motor controller in part c? Briefly explain your reasoning.

The new system has two poles at $z = \frac{3}{8}$, while the old system had one pole at $z = \frac{3}{4}$. Thus the magnitude of dominant pole in the new system is only half as big as the magnitude of the dominant pole of the old system, and the new system will respond faster than the old one.

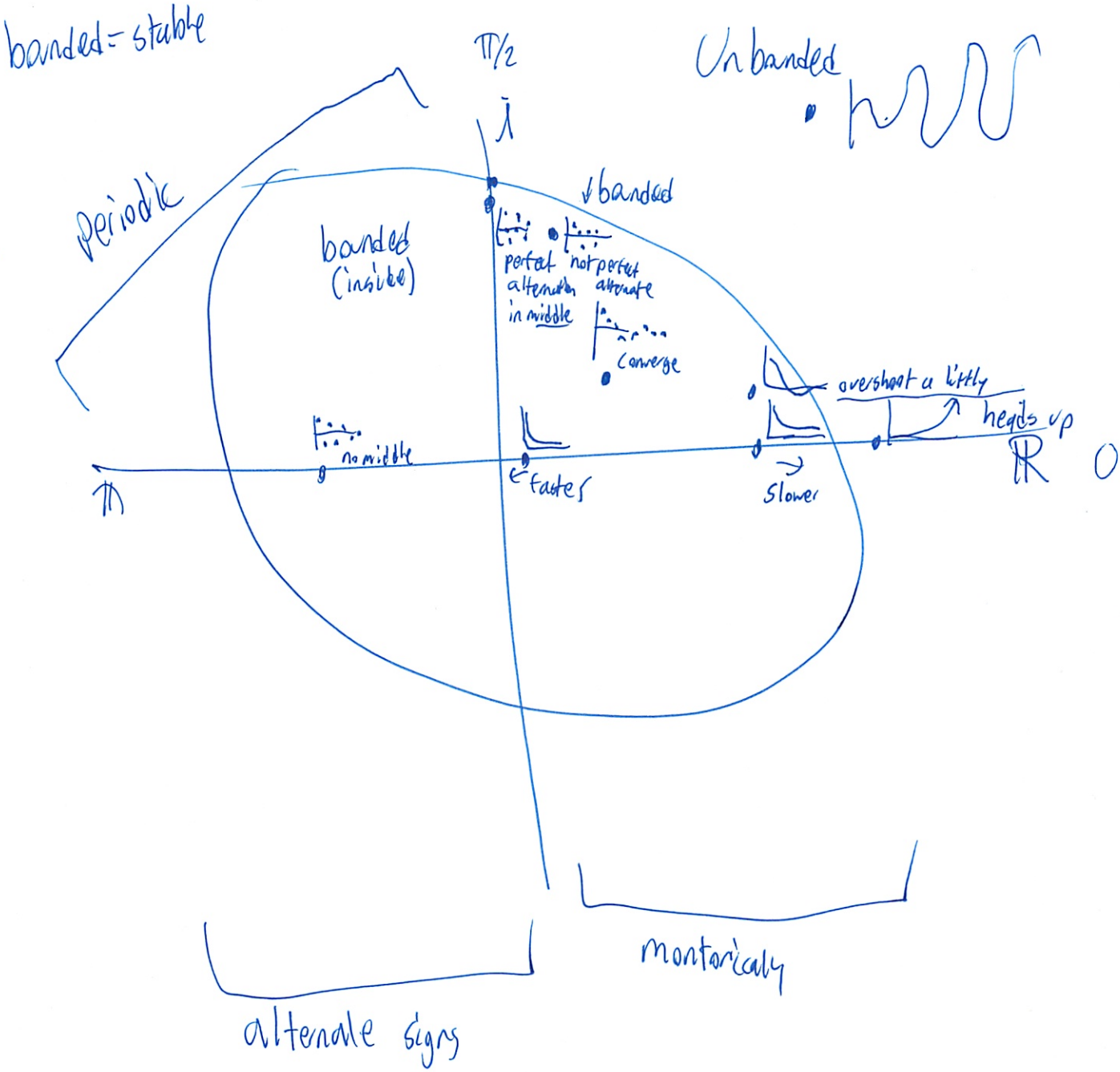
Grading: 2 points for each of parts a, b, and c; 3 points for each of parts d, e, and f.

worksheet

worksheet

12/15

Unit Circle



$$\text{angle} = \frac{2\pi}{\text{period}}$$

$$x = r\cos\theta$$
$$y = r\sin\theta$$
$$r = \sqrt{x^2+y^2}$$
$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$

Pole Cheat Sheet

6.01: Introduction to EECS I

Designing Control Systems

Week 6 *October 12, 2010*

Outline

- Complex poles
- Designing control systems

Reading: Chapter 6

Midterm exam:

- **Tonight!** 7:30–9:00 PM
- 32-141 or 32-155
- Any printed material okay
- No computers or phones
-
- No software lab today!

From last time

- Behavior of a system can be capture by its system function, which characterizes the relationship between input and output
- System functions can be combined just as PCAP modules can be combined
- Primitives are gains and delays
- Combinations include cascades, positive feedback and negative feedback
- Behavior of system captured by poles of system

Poles: Summary

- The **poles** of a system are the roots of the denominator polynomial of the system function in $1/\mathcal{R}$.
- The **dominant pole** is the pole with the largest magnitude.

Dependence on pole magnitude

Response to a bounded input signal, if the dominant pole has magnitude

- >1: output signal will be unbounded
- <1: output signal will be bounded
  if the input is transient, output signal will converge to 0.
- 1: output signal will be bounded

A system is *stable* if the output signal is bounded.

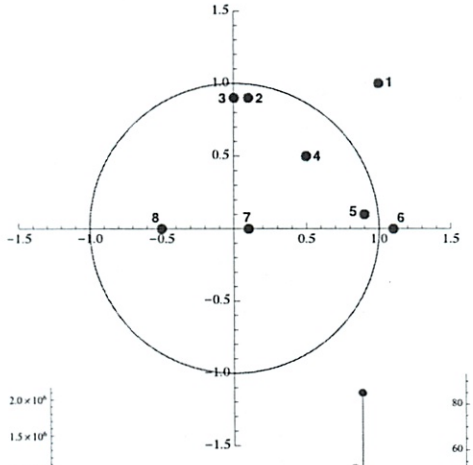
Dependence on pole type

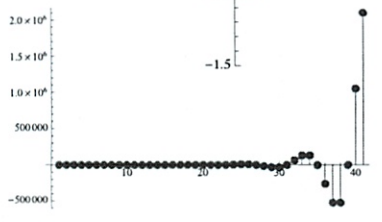
Response to a transient input signal, if the dominant pole is

- **real and positive**: output signal will, after finitely many steps, begin to increase or decrease monotonically.
- **real and negative**: output signal will, after finitely many steps, begin to alternate signs.
- **complex**: output signal will, after finitely many steps, begin to be periodic, with a period of $2\pi/\Omega$, where Ω is the 'angle' of the pole in the complex plane.

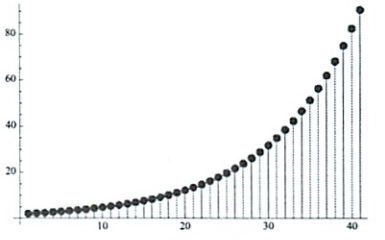
1 Pole Position (16 points)

The polar plot shows the dominant pole for several systems. Match each pole to the unit sample response of the system.

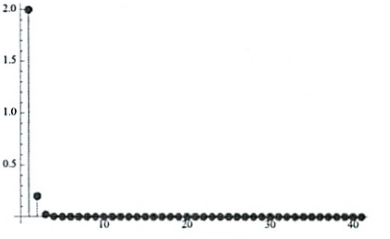




A = 1



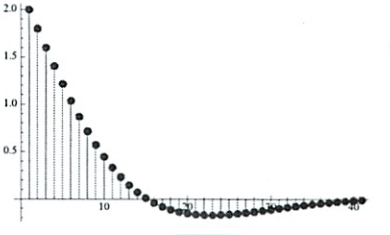
B = 6



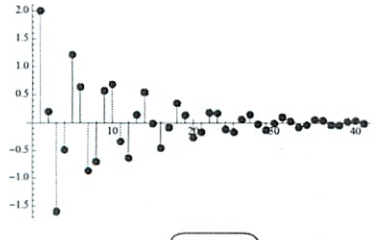
C = 7

D = 8

E = 3



F = 5



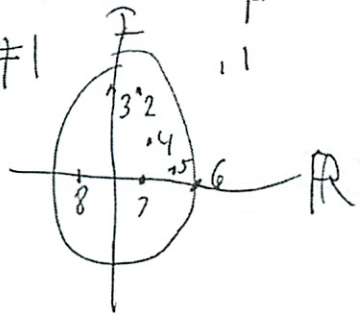
G = 2

H = 4

(2)

going to do practice problems

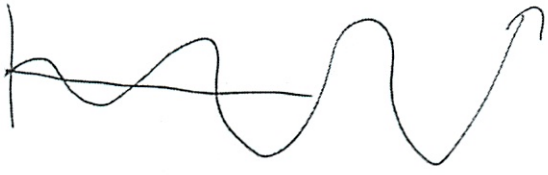
Spring10#1



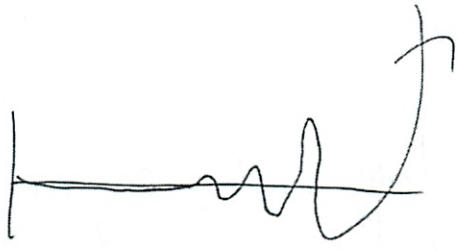
- all we care about
  - along real line
  - inside 1st quad

#1

- ossilitory divergant
  - ossilitory: has some imaginary
  - divergant: outside circle, pole may > 1



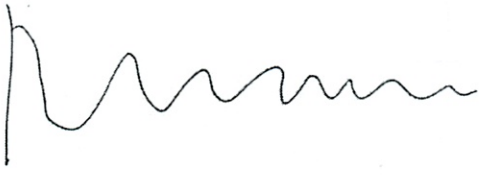
or



depends on scale of graph

#2

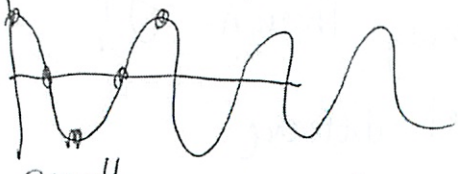
- inside, still complex
  - not divergant convergant
  - still ossilates



(but remember we are doing discritazation of it)

#3

- purly imaginary
- no real component



exactly on the line - fastest occilation at sample rate
- faster would not capture right

(3)

#4

- how #2, #4, #5 different
  - speed of/freq of oscilation
  - higher imaginary than real → faster oscilation
    oscilates medium speed

#5 oscilates very slowly

#6

outside circle, on real line
↓ ↓ ↑ no oscilate
diverges monotically

B

#7

~~this~~ on, real, inside
monotonic conversion

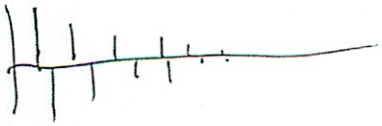
- sometimes too slow above

F

#8

differs by — sign from 7
- every other value is negitive

D



no passing through 0!
- so can tell difference

(4)

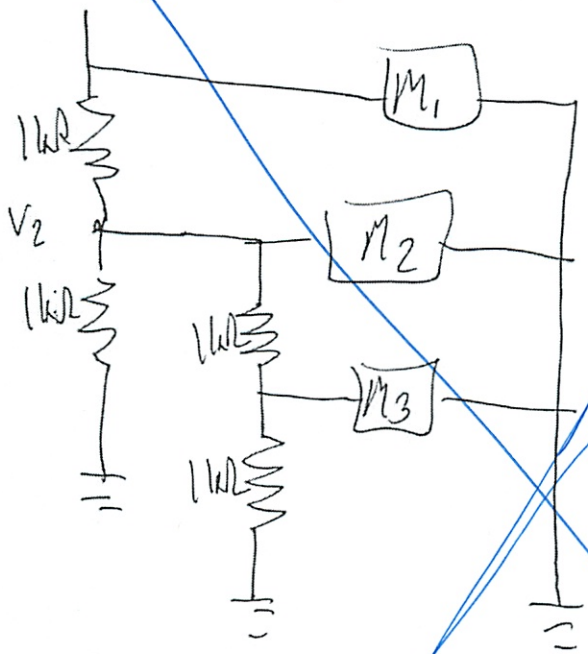
the larger the magnitude (near 1) takes a long time to converge

smaller " " (near 0) converges quickly

0 - converges immediatly

Spring 10 #4 Motor Control

- engineers have a bunch of solutions
- voltage drop across motors

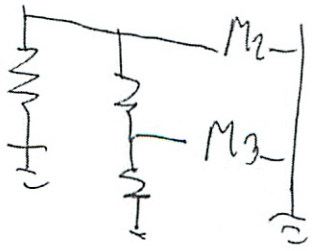


$M_1 = 10V - 0V = 10V$

$M_2 = V_2 - 0V$

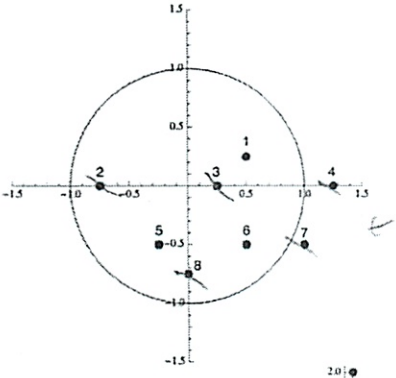
$V_2 =$ main top voltage divider but also resistors to right ground on right - so not ideal

need eq resistance

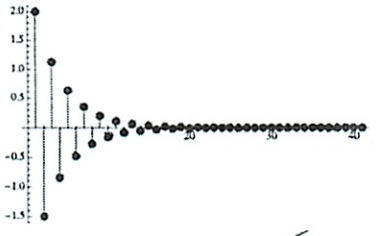


4 Pole Position (16 points)

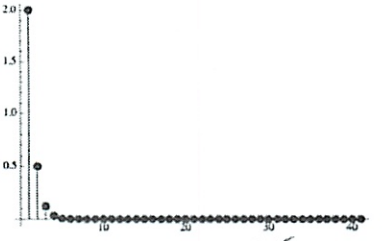
Consider eight poles located at the following locations in the z plane. The plots below show the unit-sample responses of eight linear, time-invariant systems. Match them with the dominant pole for each system (remember thta the system may have more than just one pole).



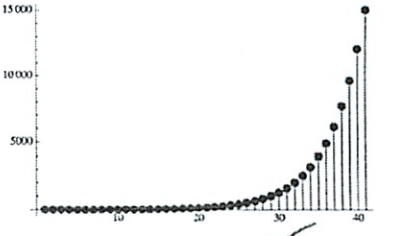
7-ossilatory divergent
4 = diverge monotonically
3 = converges monotonically
← thought we just care about 1st quad!
2 = every other value neg - no 0 slop



A = 2 ✓

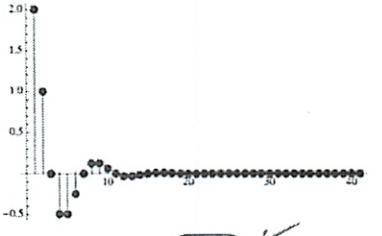


B = 3 ✓



C = 4 ✓

D = 8 ✓



E = 6 ✓

F = 5 ✓

G = 7 ✓

H = 1 ✓

?3, 1

8 = pure imaginary
oscilates perfectly
1, 5, 6 - ossilate converges
higher imag → faster ossilation
? diff 1 + 6?

6.01 Final Exam: Spring 2010

Name:	Section:

Enter all answers in the boxes provided.

During the exam you may:

- read any paper that you want to
- use a calculator

You may not

- use a computer, phone or music player

For staff use:

1.	/12
2.	/10
3.	/18
4.	/6
5.	/12
6.	/4
7.	/20
8.	/18
total:	/100

1 A Library with Class (12 points)

Let's build a class to represent a library; let's call it `Library`. In this problem, we'll deal with some standard types of objects:

- A book is represented as a string – its title.
- A patron (person who uses the library) is represented as a string – his/her name.
- A date is represented by an integer – the number of days since the library opened.

The class should have an attribute called `dailyFine` that starts out as `0.25`. The class should have the following methods:

- `__init__`: takes a list of books and initializes the library.
- `checkOut`: is given a book, a patron and a date on which the book is being checked out and it records this. Each book can be kept for 7 days before it becomes overdue, i.e. if checked out on day x, it becomes due on day $x + 7$ and it will be considered overdue by one day on day $x + 8$. It returns `None`.
- `checkIn`: is given a book and a date on which the book is being returned and it updates the records. It returns a number representing the fine due if the book is overdue and 0.0 otherwise. The fine is the number of days overdue times the value of the `dailyFine` attribute.
- `overdueBooks`: is given a patron and a date and returns the list of books which that patron has checked out which are overdue at the given date.

Here is an example of the operation of the library:

```
>>> lib = Library(['a', 'b', 'c', 'd', 'e', 'f'])
>>> lib.checkOut('a', 'T', 1)
>>> lib.checkOut('c', 'T', 1)
>>> lib.checkOut('e', 'T', 10)
>>> lib.overdueBooks('T', 13)
['a', 'c']
>>> lib.checkIn('a', 13)
1.25
>>> lib.checkIn('c', 18)
2.50
>>> lib.checkIn('e', 18)
0.25
```

In the boxes below, define the `Library` class as described above. Above each answer box we repeat the specification for each of the attributes and methods given above. Make sure that you enter complete definitions in the boxes, including complete `class` and `def` statements.

Use a dictionary to store the contents of the library. Do not repeat code if at all possible. You can assume that all the operations are legal, for example, all books checked out are in the library and books checked in have been previously checked out.

1.1

Class definition:

Include both the start of the class definition and the method definition for `__init__` in this first answer box.

The class should have an attribute called `dailyFine` that starts out as `0.25`.

`__init__`: takes a list of books and initializes the library.

`checkOut`: is given a book, a patron and a date on which the book is being checked out and it records this. Each book can be kept for 7 days before it becomes overdue, i.e. if checked out on day x, it becomes due on day $x + 7$ and it will be considered overdue by one day on day $x + 8$. It returns `None`.

`checkIn`: is given a book and a date on which the book is being returned and it updates the records. It returns a number representing the fine due if the book is overdue and 0.0 otherwise. The fine is the number of days overdue times the value of the `dailyFine` attribute.

`overdueBooks`: is given a patron and a date and returns the list of books which that patron has checked out which are overdue at the given date.

1.2

Define a new class called `LibraryGrace` that behaves just like the `Library` class except that it provides a grace period (some number of days after the actual due date) before fines start being accumulated. The number of days in the grace period is specified when an instance is created. See the example below.

```
>>> lib = LibraryGrace(2, ['a', 'b', 'c', 'd', 'e', 'f'])
>>> lib.checkOut('a', 'T', 1)
>>> lib.checkIn('a', 13)
0.75
```

Write the complete class definition for `LibraryGrace`. To get full credit you should not repeat any code that is already in the implementation of `Library`, in particular, you should not need to repeat the computation of the fine.

Class definition:

2 Library State Machine (10 points)

We will now define a state machine class, called `LibrarySM`, to operate the library.

The state machine will be initialized with a list of books and will create an instance of the `Library` class and store it as an instance variable (assume that the `Library` class is defined in the same file as your definition of `LibrarySM`).

We will allow the state machine to modify this library instance, like we did the grid instances in design lab, for the sake of efficiency. However, your `getNextValues` method should not change any other instance variables.

Each input to the state machine will be a tuple of length 2; the first element is a string indicating an operation and the second element is the "argument" for that operation. The output of the machine should be `None` unless specified otherwise below.

The allowed types of inputs (and their outputs) are illustrated by example below:

- `('day', 3)` – advance the current date by 3 (or whatever integer is the argument); the date starts at 0. Output the new date in the format `('date', 5)`.
- `('start', 'T')` – start dealing with patron `'T'`.
- `('end', 'T')` – end dealing with patron `'T'`; the output should be the total accumulated fine for that patron since the most recent `start` (which may be 0.0), in the format `('total fine', 0.5)`.
- `('co', 'a')` – check out book `'a'` for the current patron.
- `('ci', 'a')` – check in book `'a'` for the current patron; the output should be the fine if this book is overdue or 0.0 if it's not, in the format `('fine', 0.5)`.

Here is an example of the operation of the machine.

```
>>> libsm = LibrarySM(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i'])
>>> libsm.transduce([('day', 1),
                     ('start', 'T'),
                     ('co', 'a'), ('co', 'b'), ('co', 'c'),
                     ('co', 'd'), ('co', 'e'), ('co', 'f'),
                     ('end', 'T'),
                     ('start', 'X'),
                     ('co', 'g'), ('co', 'h'), ('co', 'i'),
                     ('end', 'X'),
                     ('day', 8),
                     ('start', 'T'),
                     ('ci', 'a'),
                     ('ci', 'b'),
                     ('end', 'T') ])

[('date', 1),
 None,
 None, None, None,
 None, None, None,
 ('total fine', 0.0),
 None,
 None, None, None,
 ('total fine', 0.0),
 ('date', 9),
 None,
 ('fine', 0.25),
 ('fine', 0.25),
 ('total fine', 0.5)]
```

1. Assuming that the inital date for the library is 0, what will the initial state of your machine be? Explain each component.

2. Write out the definition of `LibrarySM` class in Python. You can assume that all inputs will be legal (nobody will try to check out a book that is not in the library; there will not be a `start` for a new patron before the `end` of the previous patron; all the operations and arguments are legal, etc.).

Class definition:

Scratch paper

3 Machine Control (18 points)

Consider the following definition of a linear system, with gains `k1` and `k2`:

```
class Accumulator(sm.SM):
    startState = 0.0
    def getNextValues(self, state, inp):
        return(state+inp, state+inp)

def system(k1, k2):
    plant = Accumulator()
    sensor = sm.Delay()
    controller = sm.ParallelAdd(sm.Gain(k1),
                                sm.Cascade(Accumulator(), sm.Gain(k2)))
    return sm.FeedbackSubtract(sm.Cascade(controller, plant), sensor)
```

Note that `ParallelAdd` takes one input and provides it to two machines and outputs the **sum** of the outputs of the two machines.

Here's space to draw the block diagram, if you find it helpful (it won't be graded).

3.1

- Write a system function for the sensor.

- Write a system function for the plant.

- Write a system function for the controller, in terms of `k1` and `k2`.

- Write a system function for the cascade combination of the controller and plant, in terms of `k1` and `k2`.

- Write a system function for the whole system, in terms of `k1` and `k2`.

3.2

Imagine a different system, whose system function is given by

$$\frac{k_4\mathcal{R} - k_3 + k_3k_4}{(1-k_3)\mathcal{R}^2 + (k_3 + 3k_4 - 2)\mathcal{R} + 1}$$

If we pick $k_4 = 0$, give any non-zero value of k_3 for which the system will converge, or explain why there isn't one.

Scratch paper

4 Hot Bath (6 points)

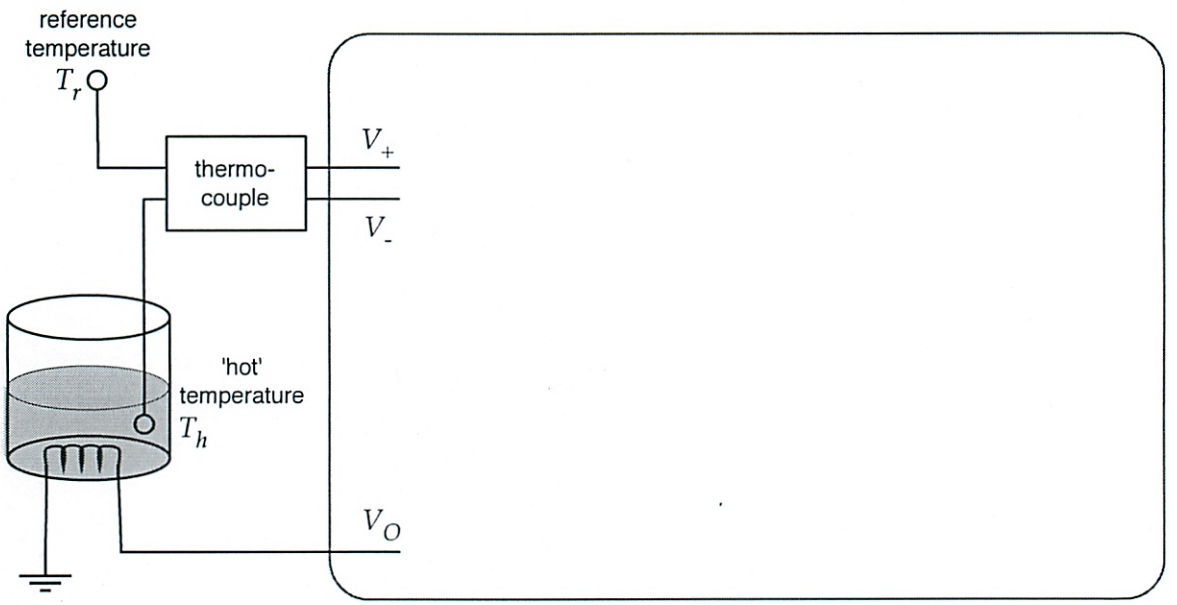
A *thermocouple* is a physical device with two temperature probes and two electronic terminals. If the probes are put in locations with different temperatures, there will be a voltage difference across the terminals. In particular,

$$V_+ - V_- = k(T_h - T_r)$$

where T_h is the temperature (°F) at the 'hot' probe, T_r is the temperature (°F) at the reference probe, and k is about 0.02.

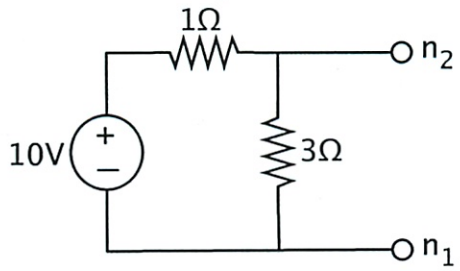
We have a vat of liquid that contains a heater (the coil at the bottom) and the 'hot' temperature sensor of the thermocouple. We would like to keep the liquid at the same temperature as the reference probe. The heater should be off if $T_r \leqslant T_h$ and be on, otherwise. When $T_r - T_h = 1°F$, then V_O should be approximately +5V.

Design a simple circuit (using one or two op-amps and some resistors of any values you want) that will achieve this; pick particular values for the resistors. Assume that we have a power supply of +10V available, and that the voltage difference $V_+ - V_-$ is in the range −10V to +10V.

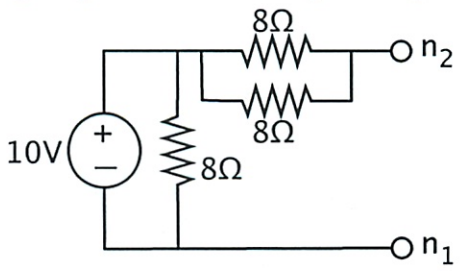


5 Equivalences (12 points)

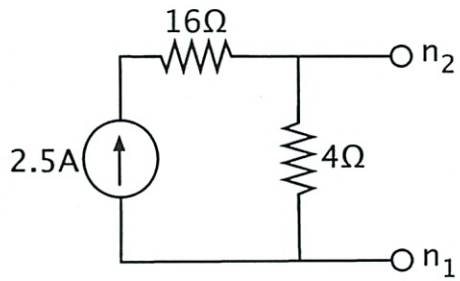
For each of the circuits below, provide the Thevenin equivalent resistance and voltage as seen from the $n_1 - n_2$ port. For the circuits with op-amps, treat them using ideal op-amp model.



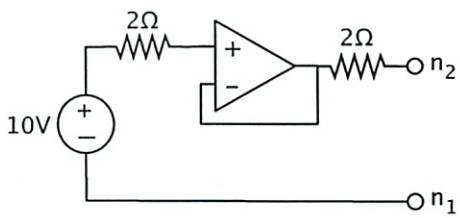
$V_{th} =$ $R_{th} =$



$V_{th} =$ $R_{th} =$



$V_{th} =$ $R_{th} =$

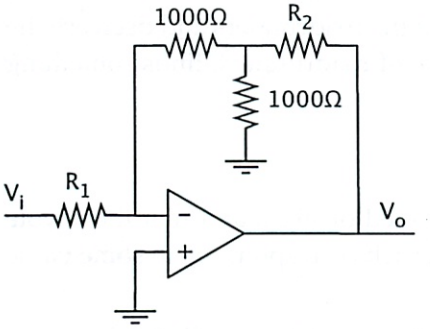


$V_{th} =$ $R_{th} =$

Scratch paper

6 T circuit (4 points)

Provide values for resistors R_1 and R_2 that make $V_o = -3V_i$.



$R_1 =$

$R_2 =$

7 Coyote v. Roadrunner (20 points)

Consider a world with some population R of roadrunners and C of coyotes. Roadrunners eat insects and coyotes eat roadrunners (when they can catch them). The roadrunner population naturally increases, but when there are coyotes around, they eat the roadrunners and decrease the roadrunner population. The coyote population, in the absence of roadrunners, finds something else to eat, and stays the same, or declines a little.

7.1 Initial distribution

Let's assume that the roadrunner population (R) can be low, med or high, and that the coyote population (C) can be low, med, or high. So, there are **9 states**, each corresponding to some value of R and some value of C.

Here is the initial **belief state**, which is written as a joint distribution over C and R, $\Pr(C, R)$.

		Coyotes		
		low	med	high
Roadrunners	low	0.04	0.20	0.18
	med	0.08	0.16	0.02
	high	0.28	0.04	0.00

1. What is the marginal distribution $\Pr(C)$?

2. What is the distribution $\Pr(R|C = low)$?

3. What is the distribution $\Pr(R|C = high)$?

4. Are R and C independent? Explain why or why not.

7.2 Transitions

Let's start by studying how the roadrunner population evolves when there are no coyotes, represented by $C_t = low$ (and the coyote population doesn't change).

$$\Pr(R_{t+1} = low \mid C_t = low, R_t = low) = 0.1$$
$$\Pr(R_{t+1} = med \mid C_t = low, R_t = low) = 0.9$$
$$\Pr(R_{t+1} = high \mid C_t = low, R_t = low) = 0.0$$
$$\Pr(R_{t+1} = low \mid C_t = low, R_t = med) = 0.0$$
$$\Pr(R_{t+1} = med \mid C_t = low, R_t = med) = 0.3$$
$$\Pr(R_{t+1} = high \mid C_t = low, R_t = med) = 0.7$$
$$\Pr(R_{t+1} = low \mid C_t = low, R_t = high) = 0.0$$
$$\Pr(R_{t+1} = med \mid C_t = low, R_t = high) = 0.0$$
$$\Pr(R_{t+1} = high \mid C_t = low, R_t = high) = 1.0$$

1. Assume $C_t = low$. For simplicity, also assume that we start out knowing with certainty that the roadrunner population is low. What is the distribution over the possible levels of the roadrunner population (low, med, high) after 1 time step?

2. Assume $C_t = low$. For simplicity, also assume that we start out knowing with certainty that the roadrunner population is low. What is the distribution over the possible levels of the roadrunner population (low, med, high) after 2 time steps?

7.3 Observations

Imagine that you are starting with the initial belief state, the joint distribution from problem 7.1. If it helps, you can think of it as the following `DDist` over pairs of values (the first is the value of R, the second is the value of C):

```
DDist({('low', 'low') : 0.04, ('low', 'med') : 0.2, ('low', 'high') : 0.18,
       ('med', 'low') : 0.08, ('med', 'med') : 0.16, ('med', 'high') : 0.02,
       ('high', 'low') : 0.28, ('high', 'med') : 0.04, ('high', 'high') : 0.00})
```

You send an ecologist out into the field to sample the numbers of roadrunners and coyotes. The ecologist can't really figure out the absolute numbers of each species, but reports one of three observations:

- **moreC**: means that there are significantly more coyotes than roadrunners (that is, that the level of coyotes is **high** and the level of roadrunners is **med** or **low**, or that the level of coyotes is **med** and the level of roadrunners is **low**).
- **moreR**: means that there are significantly more roadrunners than coyotes (that is, that the level of roadrunners is **high** and the level of coyotes is **med** or **low**, or that the level of roadrunners is **med** and the level of coyotes is **low**).
- **same**: means that there are roughly the same number of coyotes as roadrunners (the populations have the same level).

1. If there is no noise in the ecologist's observations (that is, the observation is always true, given the state), and the observation is **moreR**, what is the resulting belief state $\Pr(C_0, R_0 \mid O_0 =$ **moreR**$)$ (the distribution over states given the observation)?

2. Now, we will assume that the ecologist's observations are fallible.

$$\Pr(O_t = \textbf{moreC} \mid C_t > R_t) = 0.9$$
$$\Pr(O_t = \textbf{same} \mid C_t > R_t) = 0.1$$
$$\Pr(O_t = \textbf{moreR} \mid C_t > R_t) = 0.0$$
$$\Pr(O_t = \textbf{moreC} \mid C_t = R_t) = 0.1$$
$$\Pr(O_t = \textbf{same} \mid C_t = R_t) = 0.8$$
$$\Pr(O_t = \textbf{moreR} \mid C_t = R_t) = 0.1$$
$$\Pr(O_t = \textbf{moreC} \mid C_t < R_t) = 0.0$$
$$\Pr(O_t = \textbf{same} \mid C_t < R_t) = 0.1$$
$$\Pr(O_t = \textbf{moreR} \mid C_t < R_t) = 0.9$$

Now, if the observation is **moreR**, what is the belief state $\Pr(C_0, R_0 \mid O_0 = \textbf{moreR})$ (the distribution over states given the observation)?

8 Ab und Aufzug (18 points)

Hans, Wilhelm, and Klaus are three business tycoons in a three-story skyscraper with one elevator. We know in advance that they will call the elevator simultaneously after their meetings tomorrow and we want to get them to their destinations as quickly as possible (time is money!). We also know that the elevator will be on the first floor when they call it. We're going to use search to find the best path for the elevator to take.

Hans	starts on the 2nd floor,	and wants to go to the 1st floor.
Wilhelm	starts on the 3rd floor,	and wants to go to the 1st floor.
Klaus	starts on the 3rd floor,	and wants to go to the 2nd floor.

State will be stored as a tuple whose first element is the location of the elevator and whose second element is a tuple with the locations of Hans, Wilhelm, and Klaus, in that order. A location of `None` means that the person in question is riding the elevator. So the state

```
(2, (None, 1, 2))
```

means that Hans is on the elevator which is on the 2nd floor and Klaus is on the 2nd floor as well, but not on the elevator, whereas Wilhelm is on the 1st floor.

Our legal actions are:

```
legalActions =
  ['ElevatorDown', 'ElevatorUp', (0, 'GetsOn'), (0, 'GetsOff'),
   (1, 'GetsOn'), (1, 'GetsOff'), (2, 'GetsOn'), (2, 'GetsOff')]
```

where `"ElevatorUp"` causes the elevator to move one floor up, `"ElevatorDown"` causes it to move one floor down, and 0, 1, and 2 correspond to Hans, Wilhelm, and Klaus, respectively.

Let's say it takes one minute for the elevator to move one floor in either direction, and it takes one minute for anyone to get on or off the elevator unless Klaus is involved. It takes Klaus five minutes to get on or off the elevator (he's extremely slow); it also takes everyone else five minutes to get on or off the elevator if Klaus is already riding the elevator (he's extremely talkative).

8.1 Goal

Write the goal function.

```
def goal(state):
```

Scratch paper

8.2 Search Strategies

In what follows: ED is `'ElevatorDown'`, EU is `'ElevatorUp'`, 0On is `(0, 'GetsOn')`, etc. Also, **N** stands for `None`.

Suppose that we are in the middle of the search and the search agenda consists of the following nodes (**listed in the order that they were added to the agenda, earliest first**):

A: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{EU}} (3, (2, 3, 3)) \xrightarrow{\text{2On}} (3, (2, 3, \mathbf{N})) \xrightarrow{\text{1On}} (3, (2, \mathbf{N}, \mathbf{N}))$

B: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{0On}} (2, (\mathbf{N}, 3, 3)) \xrightarrow{\text{ED}} (1, (\mathbf{N}, 3, 3)) \xrightarrow{\text{0Off}} (1, (1, 3, 3))$

C: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{EU}} (3, (2, 3, 3)) \xrightarrow{\text{1On}} (3, (2, \mathbf{N}, 3)) \xrightarrow{\text{2On}} (3, (2, \mathbf{N}, \mathbf{N})) \xrightarrow{\text{ED}} (2, (2, \mathbf{N}, \mathbf{N}))$

- Assume that no states other than the ones listed were visited in the search.
- An illegal action leaves you in the same state and Pruning Rule 1 applies (don't consider any path that visits the same state twice).
- Assume that the order of operations is as listed at the beginning of this problem:
  ED, EU, 0On, 0Off, 1On, 1Off, 2On, 2Off

Note that in general you may have to expand more than one node to find the next state that is visited.

8.2.1 If we are doing breadth-first search (BFS)

1. Starting from this agenda, which node (A, B, or C) gets expanded first (circle one)?
   A **B** **C**
2. Which node gets added to the agenda first? Specify its parent node, action, and state.

3. What is the total path cost of the new node added to the agenda?

Agenda:

A: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{EU}} (3, (2, 3, 3)) \xrightarrow{\text{2On}} (3, (2, 3, \mathbf{N})) \xrightarrow{\text{1On}} (3, (2, \mathbf{N}, \mathbf{N}))$

B: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{0On}} (2, (\mathbf{N}, 3, 3)) \xrightarrow{\text{ED}} (1, (\mathbf{N}, 3, 3)) \xrightarrow{\text{0Off}} (1, (1, 3, 3))$

C: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{EU}} (3, (2, 3, 3)) \xrightarrow{\text{1On}} (3, (2, \mathbf{N}, 3)) \xrightarrow{\text{2On}} (3, (2, \mathbf{N}, \mathbf{N})) \xrightarrow{\text{ED}} (2, (2, \mathbf{N}, \mathbf{N}))$

8.2.2 If we are doing depth-first search (DFS)

1. Starting from this agenda, which node (A, B, or C) gets expanded first (circle one)?
   A **B** **C**
2. Which node gets added to the agenda first? Specify its parent node, action, and state.

3. What is the total path cost of the new node added to the agenda?

8.2.3 If we are doing breadth-first search with dynamic programming (BFS+DP)

1. Starting from this agenda, which node (A, B, or C) gets expanded first (circle one)?
   A **B** **C**
2. Which node gets added to the agenda first? Specify its parent node, action, and state.

3. What is the total path cost of the new node added to the agenda?

Agenda:

A: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{EU}} (3, (2, 3, 3)) \xrightarrow{\text{2On}} (3, (2, 3, \mathbf{N})) \xrightarrow{\text{1On}} (3, (2, \mathbf{N}, \mathbf{N}))$

B: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{0On}} (2, (\mathbf{N}, 3, 3)) \xrightarrow{\text{ED}} (1, (\mathbf{N}, 3, 3)) \xrightarrow{\text{0Off}} (1, (1, 3, 3))$

C: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{EU}} (3, (2, 3, 3)) \xrightarrow{\text{1On}} (3, (2, \mathbf{N}, 3)) \xrightarrow{\text{2On}} (3, (2, \mathbf{N}, \mathbf{N})) \xrightarrow{\text{ED}} (2, (2, \mathbf{N}, \mathbf{N}))$

8.2.4 If we are doing uniform-cost search (ucSearch)

1. Starting from this agenda, which node (A, B, or C) gets expanded first (circle one)?
   A **B** **C**
2. Which node gets added to the agenda first? Specify its parent node, action, and state.

3. What is the total path cost of the new node added to the agenda?

8.3 Heuristics

Frieda, Lola, Ulrike, and Xenia (four engineers) are asked to produce heuristics to speed up the search. In all the heuristics, distance is measured in number of floors.

- Frieda suggests that you use the maximum of the distances between each person and his destination plus 2 times the number of people who are not on the right floor.
- Lola suggests that you use the maximum of the distances between each person and his destination plus 10 if Klaus is not on the elevator and not on the right floor, plus 5 if Klaus is on the elevator.
- Ulrike suggests that you use the sum of the distances between each person and his destination.
- Xenia suggests that you use the maximum of the distances between each person and his destination.

Which of these heuristics are admissible? Circle Yes or No.

- **F:** Yes No
- **L:** Yes No
- **U:** Yes No
- **X:** Yes No

6.01 Final Exam: Spring 2010

Name:	Section:

Solutions: Not correct for the make-up exam.

Enter all answers in the boxes provided.

During the exam you may:

- read any paper that you want to
- use a calculator

You may not

- use a computer, phone or music player

For staff use:

1.	/12
2.	/10
3.	/18
4.	/6
5.	/12
6.	/4
7.	/20
8.	/18
total:	/100

1 A Library with Class (12 points)

Let's build a class to represent a library; let's call it `Library`. In this problem, we'll deal with some standard types of objects:

- A book is represented as a string – its title.
- A patron (person who uses the library) is represented as a string – his/her name.
- A date is represented by an integer – the number of days since the library opened.

The class should have an attribute called `dailyFine` that starts out as `0.25`. The class should have the following methods:

- `__init__`: takes a list of books and initializes the library.
- `checkOut`: is given a book, a patron and a date on which the book is being checked out and it records this. Each book can be kept for 7 days before it becomes overdue, i.e. if checked out on day x, it becomes due on day $x + 7$ and it will be considered overdue by one day on day $x + 8$. It returns `None`.
- `checkIn`: is given a book and a date on which the book is being returned and it updates the records. It returns a number representing the fine due if the book is overdue and 0.0 otherwise. The fine is the number of days overdue times the value of the `dailyFine` attribute.
- `overdueBooks`: is given a patron and a date and returns the list of books which that patron has checked out which are overdue at the given date.

Here is an example of the operation of the library:

```
>>> lib = Library(['a', 'b', 'c', 'd', 'e', 'f'])
>>> lib.checkOut('a', 'T', 1)
>>> lib.checkOut('c', 'T', 1)
>>> lib.checkOut('e', 'T', 10)
>>> lib.overdueBooks('T', 13)
['a', 'c']
>>> lib.checkIn('a', 13)
1.25
>>> lib.checkIn('c', 18)
2.50
>>> lib.checkIn('e', 18)
0.25
```

In the boxes below, define the `Library` class as described above. Above each answer box we repeat the specification for each of the attributes and methods given above. Make sure that you enter complete definitions in the boxes, including complete `class` and `def` statements.

Use a dictionary to store the contents of the library. Do not repeat code if at all possible. You can assume that all the operations are legal, for example, all books checked out are in the library and books checked in have been previously checked out.

1.1

Class definition:

Include both the start of the class definition and the method definition for `__init__` in this first answer box.

The class should have an attribute called `dailyFine` that starts out as `0.25`.

`__init__`: takes a list of books and initializes the library.

```
class Library:
    dailyFine = 0.25
    def __init__(self, books):
        self.shelf = {}
        for book in books:
            self.shelf[book] = (None, None)  # (patron, dueDate)
```

`checkOut`: is given a book, a patron and a date on which the book is being checked out and it records this. Each book can be kept for 7 days before it becomes overdue, i.e. if checked out on day x, it becomes due on day $x + 7$ and it will be considered overdue by one day on day $x + 8$. It returns `None`.

```
    def checkOut(self, book, patron, date):
        self.shelf[book] = (patron, date+7)
```


`checkIn`: is given a book and a date on which the book is being returned and it updates the records. It returns a number representing the fine due if the book is overdue and 0.0 otherwise. The fine is the number of days overdue times the value of the `dailyFine` attribute.

```
    def checkIn(self, book, date):
        patron, due = self.shelf[book]
        self.shelf[book] = (None, None)
        return max(0.0, (date - due))*self.dailyFine
```

`overdueBooks`: is given a patron and a date and returns the list of books which that patron has checked out which are overdue at the given date.

```
    def overdueBooks(self, patron, date):
        overdue = []
        for book in self.shelf:
            p, d = self.shelf[book]
            if p and d and p == patron and date > d:
                overdue.append(book)
        return overdue
```

1.2

Define a new class called `LibraryGrace` that behaves just like the `Library` class except that it provides a grace period (some number of days after the actual due date) before fines start being accumulated. The number of days in the grace period is specified when an instance is created. See the example below.

```
>>> lib = LibraryGrace(2, ['a', 'b', 'c', 'd', 'e', 'f'])
>>> lib.checkOut('a', 'T', 1)
>>> lib.checkIn('a', 13)
0.75
```

Write the complete class definition for `LibraryGrace`. To get full credit you should not repeat any code that is already in the implementation of `Library`, in particular, you should not need to repeat the computation of the fine.

Class definition:

```
class LibraryGrace(Library):
    def __init__(self, grace, books):
        self.grace = grace
        Library.__init__(self, books)
    def checkIn(self, book, date):
        return Library.checkIn(self, book, date - self.grace)
```


2 Library State Machine (10 points)

We will now define a state machine class, called `LibrarySM`, to operate the library.

The state machine will be initialized with a list of books and will create an instance of the `Library` class and store it as an instance variable (assume that the `Library` class is defined in the same file as your definition of `LibrarySM`).

We will allow the state machine to modify this library instance, like we did the grid instances in design lab, for the sake of efficiency. However, your `getNextValues` method should not change any other instance variables.

Each input to the state machine will be a tuple of length 2; the first element is a string indicating an operation and the second element is the "argument" for that operation. The output of the machine should be `None` unless specified otherwise below.

The allowed types of inputs (and their outputs) are illustrated by example below:

- `('day', 3)` – advance the current date by 3 (or whatever integer is the argument); the date starts at 0. Output the new date in the format `('date', 5)`.
- `('start', 'T')` – start dealing with patron `'T'`.
- `('end', 'T')` – end dealing with patron `'T'`; the output should be the total accumulated fine for that patron since the most recent `start` (which may be 0.0), in the format `('total fine', 0.5)`.
- `('co', 'a')` – check out book `'a'` for the current patron.
- `('ci', 'a')` – check in book `'a'` for the current patron; the output should be the fine if this book is overdue or 0.0 if it's not, in the format `('fine', 0.5)`.

Here is an example of the operation of the machine.

```
>>> libsm = LibrarySM(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i'])
>>> libsm.transduce([('day', 1),
                     ('start', 'T'),
                     ('co', 'a'), ('co', 'b'), ('co', 'c'),
                     ('co', 'd'), ('co', 'e'), ('co', 'f'),
                     ('end', 'T'),
                     ('start', 'X'),
                     ('co', 'g'), ('co', 'h'), ('co', 'i'),
                     ('end', 'X'),
                     ('day', 8),
                     ('start', 'T'),
                     ('ci', 'a'),
                     ('ci', 'b'),
                     ('end', 'T') ])
```

```
[('date', 1),
 None,
 None, None, None,
 None, None, None,
 ('total fine', 0.0),
 None,
 None, None, None,
 ('total fine', 0.0),
 ('date', 9),
 None,
 ('fine', 0.25),
 ('fine', 0.25),
 ('total fine', 0.5)]
```

1. Assuming that the inital date for the library is 0, what will the initial state of your machine be? Explain each component.

```
        self.startState = (0, None, 0.0)      # (date, patron, fine)
```


2. Write out the definition of `LibrarySM` class in Python. You can assume that all inputs will be legal (nobody will try to check out a book that is not in the library; there will not be a `start` for a new patron before the end of the previous patron; all the operations and arguments are legal, etc.).

Class definition:

```
class LibrarySM(sm.SM):
    def __init__(self, books):
        self.lib = Library(books)
        self.startState = (0, None, 0.0)      # (date, patron, fine)
    def getNextValues(self, state, inp):
        date, patron, fine = state
        operation, argument = inp
        out = None
        if operation == 'start':
            patron = argument
        elif operation == 'ci':
            newFine = self.lib.checkIn(argument, date)
            fine += newFine
            out = ('fine', newFine)
        elif operation == 'co':
            self.lib.checkOut(argument, patron, date)
        elif operation == 'day':
            date += argument
            out = ('date', date)
        elif operation == 'end':
            out = ('total fine', fine)
        else:
            print 'Illegal operation', operation
        return ((date, patron, fine), out)
```

Scratch paper

3 Machine Control (18 points)

Consider the following definition of a linear system, with gains k1 and k2:

```
class Accumulator(sm.SM):
    startState = 0.0
    def getNextValues(self, state, inp):
        return(state+inp, state+inp)

def system(k1, k2):
    plant = Accumulator()
    sensor = sm.Delay()
    controller = sm.ParallelAdd(sm.Gain(k1),
                                sm.Cascade(Accumulator(), sm.Gain(k2)))
    return sm.FeedbackSubtract(sm.Cascade(controller, plant), sensor)
```

Note that `ParallelAdd` takes one input and provides it to two machines and outputs the **sum** of the outputs of the two machines.

Here's space to draw the block diagram, if you find it helpful (it won't be graded).

3.1

- Write a system function for the sensor.

R

- Write a system function for the plant.

$$\frac{1}{1-R}$$

- Write a system function for the controller, in terms of k1 and k2.

$$\frac{k_1(1-R)+k2}{1-R}$$

- Write a system function for the cascade combination of the controller and plant, in terms of k1 and k2.

$$\frac{k_1(1-R)+k2}{1-R}\,\frac{1}{1-R} = \frac{k_1(1-R)+k2}{(1-R)^2}$$

- Write a system function for the whole system, in terms of k1 and k2.

$$\frac{-k_1R+k_1+k_2}{(1-k_1)R^2+(k_1+k_2-2)R+1}$$

3.2

Imagine a different system, whose system function is given by

$$\frac{k_4\mathcal{R}-k_3+k_3k_4}{(1-k_3)\mathcal{R}^2+(k_3+3k_4-2)\mathcal{R}+1}$$

If we pick $k_4 = 0$, give any non-zero value of k_3 for which the system will converge, or explain why there isn't one.

With $k_4 = 0$, the denominator of the system function is:

$$(1-k_3)R^2+(k_3-2)R+1$$

The roots of the corresponding z polynomial are $\{1, 1-k_3\}$. The pole at 1 means that the system has a non-decreasing oscillation.

Scratch paper

4 Hot Bath (6 points)

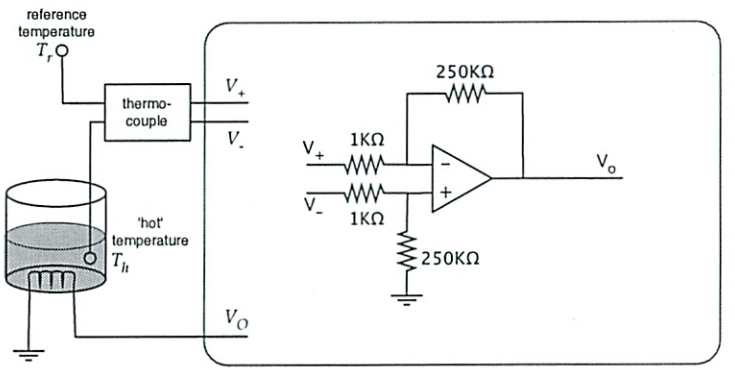
A *thermocouple* is a physical device with two temperature probes and two electronic terminals. If the probes are put in locations with different temperatures, there will be a voltage difference across the terminals. In particular,

$$V_+ - V_- = k(T_h - T_r)$$

where T_h is the temperature (°F) at the 'hot' probe, T_r is the temperature (°F) at the reference probe, and k is about 0.02.

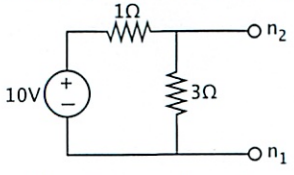
We have a vat of liquid that contains a heater (the coil at the bottom) and the 'hot' temperature sensor of the thermocouple. We would like to keep the liquid at the same temperature as the reference probe. The heater should be off if $T_r \leqslant T_h$ and be on, otherwise. When $T_r - T_h = 1°F$, then V_O should be approximately +5V.

Design a simple circuit (using one or two op-amps and some resistors of any values you want) that will achieve this; pick particular values for the resistors. Assume that we have a power supply of +10V available, and that the voltage difference $V_+ - V_-$ is in the range −10V to +10V.

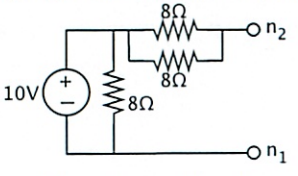


5 Equivalences (12 points)

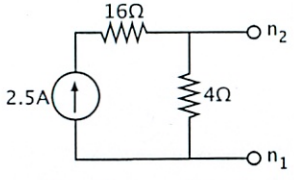
For each of the circuits below, provide the Thevenin equivalent resistance and voltage as seen from the $n_1 - n_2$ port. For the circuits with op-amps, treat them using ideal op-amp model.



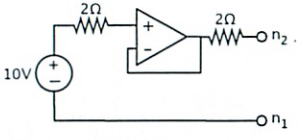
$V_{th} = 7.5$ $R_{th} = 0.75$



$V_{th} = 10$ $R_{th} = 4$



$V_{th} = 10$ $R_{th} = 4$

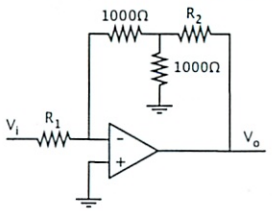


$V_{th} = 10$ $R_{th} = 2$

Scratch paper

6 T circuit (4 points)

Provide values for resistors R_1 and R_2 that make $V_o = -3V_i$.



$R_1 = 1000\Omega$

$R_2 = 1000\Omega$

7 Coyote v. Roadrunner (20 points)

Consider a world with some population R of roadrunners and C of coyotes. Roadrunners eat insects and coyotes eat roadrunners (when they can catch them). The roadrunner population naturally increases, but when there are coyotes around, they eat the roadrunners and decrease the roadrunner population. The coyote population, in the absence of roadrunners, finds something else to eat, and stays the same, or declines a little.

7.1 Initial distribution

Let's assume that the roadrunner population (R) can be low, med or high, and that the coyote population (C) can be low, med, or high. So, there are **9 states**, each corresponding to some value of R and some value of C.

Here is the initial **belief state**, which is written as a joint distribution over C and R, Pr(C, R).

		Coyotes		
		low	med	high
Roadrunners	low	0.04	0.20	0.18
	med	0.08	0.16	0.02
	high	0.28	0.04	0.00

1. What is the marginal distribution Pr(C)?

$Pr(C) = \{0.4, 0.4, 0.2\}$

2. What is the distribution Pr(R|C = low)?

$Pr(R|C = low) = \{0.04/0.4, 0.08/0.4, 0.28/0.4\} = \{0.1, 0.2, 0.7\}$

3. What is the distribution Pr(R|C = high)?

$Pr(R|C = high) = \{0.18/0.2, 0.02/0.2, 0.0/0.2\} = \{0.9, 0.1, 0.0\}$

4. Are R and C independent? Explain why or why not.

No, if they were independent, we would have, for example,

$$Pr(R = high, C = high) = Pr(R = high)P(C = high)$$

and it isn't.

7.2 Transitions

Let's start by studying how the roadrunner population evolves when there are no coyotes, represented by $C_t = \text{low}$ (and the coyote population doesn't change).

$$\Pr(R_{t+1} = \text{low} \mid C_t = \text{low}, R_t = \text{low}) = 0.1$$
$$\Pr(R_{t+1} = \text{med} \mid C_t = \text{low}, R_t = \text{low}) = 0.9$$
$$\Pr(R_{t+1} = \text{high} \mid C_t = \text{low}, R_t = \text{low}) = 0.0$$
$$\Pr(R_{t+1} = \text{low} \mid C_t = \text{low}, R_t = \text{med}) = 0.0$$
$$\Pr(R_{t+1} = \text{med} \mid C_t = \text{low}, R_t = \text{med}) = 0.3$$
$$\Pr(R_{t+1} = \text{high} \mid C_t = \text{low}, R_t = \text{med}) = 0.7$$
$$\Pr(R_{t+1} = \text{low} \mid C_t = \text{low}, R_t = \text{high}) = 0.0$$
$$\Pr(R_{t+1} = \text{med} \mid C_t = \text{low}, R_t = \text{high}) = 0.0$$
$$\Pr(R_{t+1} = \text{high} \mid C_t = \text{low}, R_t = \text{high}) = 1.0$$

1. Assume $C_t = \text{low}$. For simplicity, also assume that we start out knowing with certainty that the roadrunner population is low. What is the distribution over the possible levels of the roadrunner population (low, med, high) after 1 time step?

$\Pr(R_1 \mid R_0 = \text{low}, C_0 = \text{low}) = \langle 0.1, 0.9, 0.0 \rangle$

2. Assume $C_t = \text{low}$. For simplicity, also assume that we start out knowing with certainty that the roadrunner population is low. What is the distribution over the possible levels of the roadrunner population (low, med, high) after 2 time steps?

$\Pr(R_2 \mid R_0 = \text{low}, C_0 = \text{low}) = \langle 0.01, 0.09 + 0.24, 0.63 \rangle = \langle 0.01, 0.36, 0.63 \rangle$

7.3 Observations

Imagine that you are starting with the initial belief state, the joint distribution from problem 7.1. If it helps, you can think of it as the following `DDist` over pairs of values (the first is the value of R, the second is the value of C):

```
DDist({('low', 'low') : 0.04, ('low', 'med') : 0.2, ('low', 'high') : 0.18,
       ('med', 'low') : 0.08, ('med', 'med') : 0.16, ('med', 'high') : 0.02,
       ('high', 'low') : 0.28, ('high', 'med') : 0.04, ('high', 'high') : 0.00})
```

You send an ecologist out into the field to sample the numbers of roadrunners and coyotes. The ecologist can't really figure out the absolute numbers of each species, but reports one of three observations:

- **moreC**: means that there are significantly more coyotes than roadrunners (that is, that the level of coyotes is **high** and the level of roadrunners is **med** or **low**, or that the level of coyotes is **med** and the level of roadrunners is **low**).
- **moreR**: means that there are significantly more roadrunners than coyotes (that is, that the level of roadrunners is **high** and the level of coyotes is **med** or **low**, or that the level of roadrunners is **med** and the level of coyotes is **low**).
- **same**: means that there are roughly the same number of coyotes as roadrunners (the populations have the same level).

1. If there is no noise in the ecologist's observations (that is, the observation is always true, given the state), and the observation is **moreR**, what is the resulting belief state $\Pr(C_0, R_0 \mid O_0 = \textbf{moreR})$ (the distribution over states given the observation)?

		Coyotes		
		low	med	high
Roadrunners	low	0.0	0.00	0.00
	med	0.2	0.0	0.00
	high	0.7	0.1	0.00

2. Now, we will assume that the ecologist's observations are fallible.

$$\Pr(O_t = \textbf{moreC} \mid C_t > R_t) = 0.9$$
$$\Pr(O_t = \textbf{same} \mid C_t > R_t) = 0.1$$
$$\Pr(O_t = \textbf{moreR} \mid C_t > R_t) = 0.0$$
$$\Pr(O_t = \textbf{moreC} \mid C_t = R_t) = 0.1$$
$$\Pr(O_t = \textbf{same} \mid C_t = R_t) = 0.8$$
$$\Pr(O_t = \textbf{moreR} \mid C_t = R_t) = 0.1$$
$$\Pr(O_t = \textbf{moreC} \mid C_t < R_t) = 0.0$$
$$\Pr(O_t = \textbf{same} \mid C_t < R_t) = 0.1$$
$$\Pr(O_t = \textbf{moreR} \mid C_t < R_t) = 0.9$$

Now, if the observation is **moreR**, what is the belief state $\Pr(C_0, R_0 \mid O_0 = \textbf{moreR})$ (the distribution over states given the observation)?

		Coyotes		
		low	med	high
Roadrunners	low	0.004/0.38 = 0.011	0.00	0.00
	med	0.072/0.38 = 0.189	0.016/0.38 = 0.042	0.00
	high	0.252/0.38 = 0.663	0.036/0.38 = 0.094	0.00

8 Ab und Aufzug (18 points)

Hans, Wilhelm, and Klaus are three business tycoons in a three-story skyscraper with one elevator. We know in advance that they will call the elevator simultaneously after their meetings tomorrow and we want to get them to their destinations as quickly as possible (time is money!). We also know that the elevator will be on the first floor when they call it. We're going to use search to find the best path for the elevator to take.

Hans	starts on the 2nd floor,	and wants to go to the 1st floor.
Wilhelm	starts on the 3rd floor,	and wants to go to the 1st floor.
Klaus	starts on the 3rd floor,	and wants to go to the 2nd floor.

State will be stored as a tuple whose first element is the location of the elevator and whose second element is a tuple with the locations of Hans, Wilhelm, and Klaus, in that order. A location of `None` means that the person in question is riding the elevator. So the state

```
(2, (None, 1, 2))
```

means that Hans is on the elevator which is on the 2nd floor and Klaus is on the 2nd floor as well, but not on the elevator, whereas Wilhelm is on the 1st floor.

Our legal actions are:

```
legalActions =
  ['ElevatorDown', 'ElevatorUp', (0, 'GetsOn'), (0, 'GetsOff'),
   (1, 'GetsOn'), (1, 'GetsOff'), (2, 'GetsOn'), (2, 'GetsOff')]
```

where `"ElevatorUp"` causes the elevator to move one floor up, `"ElevatorDown"` causes it to move one floor down, and 0, 1, and 2 correspond to Hans, Wilhelm, and Klaus, respectively.

Let's say it takes one minute for the elevator to move one floor in either direction, and it takes one minute for anyone to get on or off the elevator unless Klaus is involved. It takes Klaus five minutes to get on or off the elevator (he's extremely slow); it also takes everyone else five minutes to get on or off the elevator if Klaus is already riding the elevator (he's extremely talkative).

8.1 Goal

Write the goal function.

```
def goal(state):

  return state[1] == (1, 1, 2)
```

Scratch paper

8.2 Search Strategies

In what follows: `ED` is `'ElevatorDown'`, `EU` is `'ElevatorUp'`, `0On` is `(0, 'GetsOn')`, etc. Also, N stands for `None`.

Suppose that we are in the middle of the search and the search agenda consists of the following nodes **(listed in the order that they were added to the agenda, earliest first)**:

A: $(1, (2, 3, 3)) \xrightarrow{EU} (2, (2, 3, 3)) \xrightarrow{EU} (3, (2, 3, 3)) \xrightarrow{2On} (3, (2, 3, N)) \xrightarrow{1On} (3, (2, N, N))$

B: $(1, (2, 3, 3)) \xrightarrow{EU} (2, (2, 3, 3)) \xrightarrow{0On} (2, (N, 3, 3)) \xrightarrow{ED} (1, (N, 3, 3)) \xrightarrow{0Off} (1, (1, 3, 3))$

C: $(1, (2, 3, 3)) \xrightarrow{EU} (2, (2, 3, 3)) \xrightarrow{EU} (3, (2, 3, 3)) \xrightarrow{1On} (3, (2, N, 3)) \xrightarrow{2On} (3, (2, N, N)) \xrightarrow{ED} (2, (2, N, N))$

- Assume that no states other than the ones listed were visited in the search.
- An illegal action leaves you in the same state and Pruning Rule 1 applies (don't consider any path that visits the same state twice).
- Assume that the order of operations is as listed at the beginning of this problem:
  ED, EU, 0On, 0Off, 1On, 1Off, 2On, 2Off

Note that in general you may have to expand more than one node to find the next state that is visited.

8.2.1 If we are doing breadth-first search (BFS)

1. Starting from this agenda, which node (A, B, or C) gets expanded first (circle one)?
   A* **B** **C**
2. Which node gets added to the agenda first? Specify its parent node, action, and state.

   A, ElevatorDown, (2, (2, N, N))

3. What is the total path cost of the new node added to the agenda?

   13 minutes

Agenda:

A: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{EU}} (3, (2, 3, 3)) \xrightarrow{\text{2On}} (3, (2, 3, \text{N})) \xrightarrow{\text{1On}} (3, (2, \text{N}, \text{N}))$

B: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{0On}} (2, (\text{N}, 3, 3)) \xrightarrow{\text{ED}} (1, (\text{N}, 3, 3)) \xrightarrow{\text{0Off}} (1, (1, 3, 3))$

C: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{EU}} (3, (2, 3, 3)) \xrightarrow{\text{1On}} (3, (2, \text{N}, 3)) \xrightarrow{\text{2On}} (3, (2, \text{N}, \text{N})) \xrightarrow{\text{ED}} (2, (2, \text{N}, \text{N}))$

8.2.2 If we are doing depth-first search (DFS)

1. Starting from this agenda, which node (A, B, or C) gets expanded first (circle one)?
   A B C*
2. Which node gets added to the agenda first? Specify its parent node, action, and state.

   C, ElevatorDown, (1, (2, N, N))
3. What is the total path cost of the new node added to the agenda?

   10 minutes

8.2.3 If we are doing breadth-first search with dynamic programming (BFS+DP)

1. Starting from this agenda, which node (A, B, or C) gets expanded first (circle one)?
   A* B C
2. Which node gets added to the agenda first? Specify its parent node, action, and state.

   B, ElevatorUp, (2, (1, 3, 3))
3. What is the total path cost of the new node added to the agenda?

   5 minutes

Agenda:

A: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{EU}} (3, (2, 3, 3)) \xrightarrow{\text{2On}} (3, (2, 3, \text{N})) \xrightarrow{\text{1On}} (3, (2, \text{N}, \text{N}))$

B: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{0On}} (2, (\text{N}, 3, 3)) \xrightarrow{\text{ED}} (1, (\text{N}, 3, 3)) \xrightarrow{\text{0Off}} (1, (1, 3, 3))$

C: $(1, (2, 3, 3)) \xrightarrow{\text{EU}} (2, (2, 3, 3)) \xrightarrow{\text{EU}} (3, (2, 3, 3)) \xrightarrow{\text{1On}} (3, (2, \text{N}, 3)) \xrightarrow{\text{2On}} (3, (2, \text{N}, \text{N})) \xrightarrow{\text{ED}} (2, (2, \text{N}, \text{N}))$

8.2.4 If we are doing uniform-cost search (ucSearch)

1. Starting from this agenda, which node (A, B, or C) gets expanded first (circle one)?
   A B* C
2. Which node gets added to the agenda first? Specify its parent node, action, and state.

   B, ElevatorUp, (2, (1, 3, 3))
3. What is the total path cost of the new node added to the agenda?

   5 minutes

8.3 Heuristics

Frieda, Lola, Ulrike, and Xenia (four engineers) are asked to produce heuristics to speed up the search. In all the heuristics, distance is measured in number of floors.

- Frieda suggests that you use the maximum of the distances between each person and his destination plus 2 times the number of people who are not on the right floor.
- Lola suggests that you use the maximum of the distances between each person and his destination plus 10 if Klaus is not on the elevator and not on the right floor, plus 5 if Klaus is on the elevator.
- Ulrike suggests that you use the sum of the distances between each person and his destination.
- Xenia suggests that you use the maximum of the distances between each person and his destination.

Which of these heuristics are admissible? Circle Yes or No.

- F: Yes No
- L: Yes No
- U: Yes No
- X: Yes No

L and X are both admissible; F and U are both inadmissible.

State Estimation

Now, let's assume a particular environment for the cat and mouse, and that the cat doesn't know exactly where the mouse is. The cat and mouse live in a world that is a 1 by 10 grid. On each step, the mouse moves to one of the two neighboring (E, W) grid squares uniformly at random, unless it is currently at location 0 or 9, in which case it stays where it is with probability 0.5 and moves to the neighboring square with probability 0.5.

The cat always knows where it is in this world, but not where the mouse is.

The cat's actions are to move east and west, and they always have the intended effect (except when they would cause it to move off the edge of the world, in which case they have no effect). The observation is the result of listening; on each step, the cat will "hear" a distance between 0 and 10. If it is 0, then the mouse is on the same square as the cat. If it is 1, then it is one square away; if 2, two squares, away; etc. If, for example, the cat is on location 1 and it "hears" a distance of 5,

Adv. Practice Problems

then the mouse *must* be on location 6 (because for it to be 5 squares away in the other direction, it would have to be off the end of the world.)

Question 19: Imagine that cat starts at location 5 and observes that the mouse is 3 squares away. What is the cat's belief state about the location of the mouse?

Question 20: Now, let the cat move east (and the mouse moves as described above). Before making any observation, what is the state of the system? (List both the cat's location and the belief state about the mouse's location).

Question 21: If the cat now observes that the mouse is 3 squares away, what is the new belief state about the location of the mouse?

Question 22: If, instead, the cat observes that the mouse is 5 squares away (after the transition in question 20), what is the new belief state about the location of the mouse?

Question 23: Now, we'll construct a class that can do belief state estimation for this problem. The class definition and the first line of the initialization method are provided below.

```
class CatMouseState():
    def __init__(self, catLoc):
```

Write the rest of the initialization method, which takes as input the known location of the cat, and defines two instance variables: `self.catLocation`, which contains an integer from 0 to 9 indicating the cat's location; and `self.mouseBelief` which contains a list of 10 probabilities indicating the cat's belief about the mouse's location. The initial value of `self.mouseBelief` should be a representation of the distribution corresponding to having no information about the location of the mouse.

Question 24: Provide the bodies of the following methods for the `CatMouseState` class. The procedures should modify the components of the object, and need not return any values. `actionUpdate` should update the state (cat's location and mouse belief state) given the cat's action (but remember that the mouse also moves), and `observationUpdate` should update the belief state based on its observation of the mouse.

```
#  Cat's action can be 'E' or 'W'
def actionUpdate(self, action):
```

```
# Observation: distance to mouse, as described above.
def observationUpdate(self, obs):
```

5 Friend or Foe (15 points)

Imagine that you are defending a city and there is an aircraft flying toward you. It may be important to know whether that aircraft is a 'friend' or a 'foe' (enemy). Assume you have a radar sensor that can give you noisy information about the type of the aircraft that is approaching.

We will model this situation as an HMM, in which:

- The state space is described by two components `d` and `a`, where `d` is the number of miles (`0, 1, ..., 10`) away the target is and `a` is its *attitude* to you, which is either `'friend'` or `'foe'`. The values of `d` correspond to ranges of distance, so that, in fact, `d == 0` means the aircraft is somewhere between 0 and 1 miles away, etc.
- The observation space is `{'oFriend', 'oFoe'}`, which stands for *observed friend* and *observed foe*.
- There are no actions (or, if you prefer, a single action, which just waits a time step).
- The transition model is specified in the following Python method, which takes a state `s` as input and returns a `DDist` over possible next states:

```
def transitionModel(s, i):
    # note that the i (the input action) is ignored
    (d, a) = s
    if d == 0:
        return DDist({(0, a): 1.0})
    else
        return DDist({(d, a): 0.5, (d-1, a): 0.5})
```

- The observation model is as described in the following Python method, which takes a state `s` as input and returns a `DDist` over possible observations:

```
def observationModel(s):
    (d, a) = s
    if a == 'friend':
        return DDist({'oFriend': 1 - d / 20.0, 'oFoe' : d / 20.0})
    else:
        return DDist({'oFoe': 1 - d / 20.0, 'oFriend' : d / 20.0})
```

Questions:

1. Assume that the aircraft are approaching with a constant velocity. What velocity, in miles per time step, would generate the transition model over distance intervals given in the transition model?

   Answer: 

2. Provide an alternative transition model in which friendly aircraft move 1.5 miles per time step (and the foes move at the same rate as the given model.)

3. At what distance is our sensor most useful?

   Answer:

   How does it behave at that distance?

   Answer:

4. Using the original transition and observation models, if the initial belief state, b_0, is

   `DDist({(10, 'friend') : 0.5, (10, 'foe'): 0.5})`

   then what would b_0' be, after receiving observation o_0 =`'oFriend'`.

   Answer:

5. After that, what would the belief state b_1 be?

   Answer:

6. How many non-zero entries are there in b_4 (the answer will be the same for any sequence of observations)?

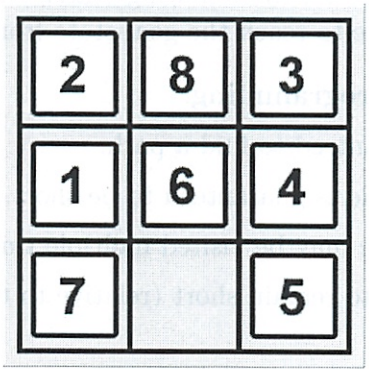
   Answer:

7. Starting again from the initial belief state, and imagining the following observation sequence: `['oFriend', 'oFoe', 'oFriend', 'oFoe']`, is it more likely that the aircraft is a friend or a foe?

   Answer:

7 A Puzzle (20 points)

Consider the standard *Eight Puzzle*. It has 8 tiles arranged on a 3 by 3 grid. Here is one possible arrangement of the tiles:



The tiles neighboring an empty space can be slid into the space. We can think of the operations on the puzzle as *moving the space* **up**, **down**, **left**, or **right**. It obviously cannot be moved beyond the bounds of the puzzle. In the example above, if we were to slide the space **up**, then the **6** tile would be in the bottom row and the space would be in the middle.

We can solve this problem using search. A state of the search would be an array of numbers (and None for the space) showing the location of each tile on grid. A goal state would be some specified arrangement of the tiles.

Assuming we apply pruning rule 1, but not the others, how many descendants are there for the state shown above:

- at level 1 (immediate children)?

  Answer: ____

- at level 2 (children of level 1)?

  Answer: ____

7.1 Search

We would like to identify the advantages and disadvantages of each of the following search methods **for this problem**. We assume, as always, that we use pruning rules 1 and 2.

Enter T or F in the boxes provided if the following statement is true or false, respectively. You can assume that it is possible to reach the goal state from the initial state.

- **depth-first, no dynamic programming:**
  - ☐ This method is guaranteed to find a path.
  - ☐ The path that is found is guaranteed to be short.
  - ☐ The same board state may be visited multiple times.
  - ☐ The agenda is likely to remain short (relative to the other methods).

- **depth-first, with dynamic programming:**
  - ☐ This method is guaranteed to find a path.
  - ☐ The path that is found is guaranteed to be short.
  - ☐ The same board state may be visited multiple times.
  - ☐ The agenda is likely to remain short (relative to the other methods).

- **breadth-first, no dynamic programming:**
  - ☐ This method is guaranteed to find a path.
  - ☐ The path that is found is guaranteed to be short.
  - ☐ The same board state may be visited multiple times.
  - ☐ The agenda is likely to remain short (relative to the other methods).

- **breadth-first, with dynamic programming:**
  - ☐ This method is guaranteed to find a path.
  - ☐ The path that is found is guaranteed to be short.
  - ☐ The same board state may be visited multiple times.
  - ☐ The agenda is likely to remain short (relative to the other methods).

7.2 State machine

We can describe the puzzle and its evolution using a state machine, in much the same way we described the wolf-goat-cabbage problem. We will specify the state with two components:

- A pair of *row*, *column* indices, indicating where the space is; and
- A list of three lists of items; each item is a digit between 1 and 8, or `None`. This describes the state of the board.

So, for example, we could represent the puzzle state above as:

```
((2, 1), [[2, 8, 3], [1, 6, 4], [7, None, 5]])
```

Technically speaking, we don't need the first component of the state (it could always be computed from the list of three lists), but it will simplify our coding if we maintain both representations.

A skeleton of the state machine defining the eight puzzle is shown on the next page. Fill in the definition of `getNextValues`. If a move would produce and illegal state, then the new state should be the same as the current state. The output of the machine should just be the next state.

```
class EightPuzzle(SM):
    startState = ((2, 1), [[2, 8, 3], [1, 6, 4], [7, None, 5]])
    size = 3
    offsets = {'up': (-1, 0), 'down': (1, 0), 'left': (0, -1), 'right': (0, 1)}
    legalInputs = ['up', 'down', 'left', 'right']

    def getNextValues(self, state, inp):
        ((p1, p2), board) = state
        bcopy = copy.deepcopy(board)
```

6.01 Day Before

12/15

Make up final

our base op amp = voltage follower

remember state - as small as possible b/c will search through it

Admissible heuristic - under

(ember open book - just practice)

Do past exam
T+W problems