

6.01 Midterm 2: Spring 2009

Name:

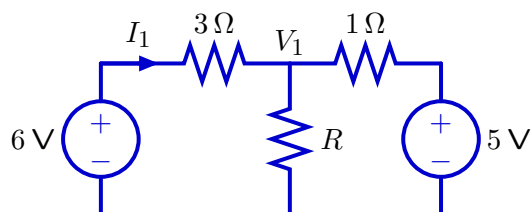
Enter all answers in the boxes provided.
You may use any paper materials you have brought.
No computers, cell phones, or music players.

For staff use:

1.	/20
2.	/20
3.	/20
4.	/20
5.	/20
total:	/100

1 Circuits (20 points)

Consider the following circuit where the resistance R is in the range $0 \leq R \leq \infty$.



Part a. Determine I_1 if $R = 0\Omega$.

$$I_1 = \boxed{2\text{ A}}$$

$$I_1 = \frac{6\text{ V}}{3\Omega} = 2\text{ A}$$

Part b. Determine V_1 if $R = 1\Omega$.

$$V_1 = \boxed{3\text{ V}}$$

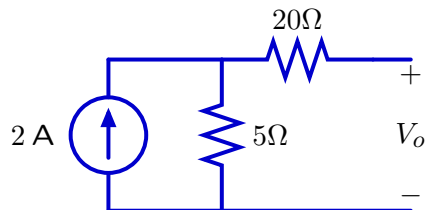
$$\frac{V_1 - 6}{3} + \frac{V_1}{1} + \frac{V_1 - 5}{1} = 0$$

$$\frac{7}{3}V_1 = 7$$

$$V_1 = 3$$

2 Equivalent Circuits (20 points)

Part a. Determine the Thevenin equivalent voltage V_{TH} (also called the open circuit voltage) and Thevenin equivalent resistance R_{TH} of the following circuit, viewed from the V_o port.



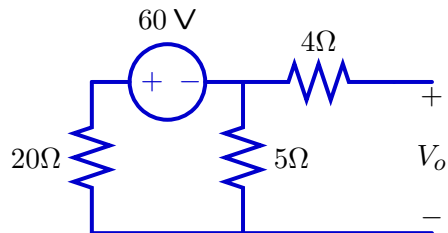
V_{TH} :

10 V

R_{TH} :

25 Ω

Part b. Determine the Thevenin equivalent voltage V_{TH} (also called the open circuit voltage) and Thevenin equivalent resistance R_{TH} of the following circuit, viewed from the V_o port.

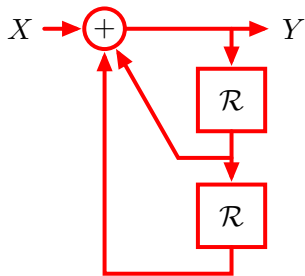
 $V_{TH}:$ -12 V $R_{TH}:$ $8\ \Omega$

3 State Machine Behaviors (20 points)

Consider the following state machine.

```
class mySystem(sm.SM):
    startState = (0,0)
    def getNextValues(self, state, inp):
        (y1,y2) = state
        y0 = inp + y1 + y2
        return ((y0,y1), y0)
```

Part a. An instance of `mySystem` can be represented by a block diagram that contains only adders, gains, and/or delays. Draw this block diagram in the space below.



Part b. Determine the result of the following Python expression.

```
mySystem().transduce([1,0,0,0,0,0,0,0])
```

Enter the result in the box below.

[1, 1, 2, 3, 5, 8, 13, 21]

Part c. Determine the magnitude of the dominant pole of the system represented by `mySystem()`, and enter it in the box below.

magnitude of dominant pole:

$$\frac{1 + \sqrt{5}}{2}$$

$$H = \frac{Y}{X} = \frac{1}{1 - \mathcal{R} - \mathcal{R}^2} = \frac{1}{1 - \frac{1}{z} - \frac{1}{z^2}} = \frac{z^2}{z^2 - z - 1}$$

$$\text{poles are at } \frac{1}{2} \pm \sqrt{\left(\frac{1}{2}\right)^2 + 1} = \frac{1}{2} \pm \sqrt{\frac{5}{4}} = \frac{1 \pm \sqrt{5}}{2}$$

Part d. In the space below, write a new subclass of `sm.SM` called `newSystem`. Instances of `newSystem` should have a system function H given by

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

```
class newSystem(sm.SM):

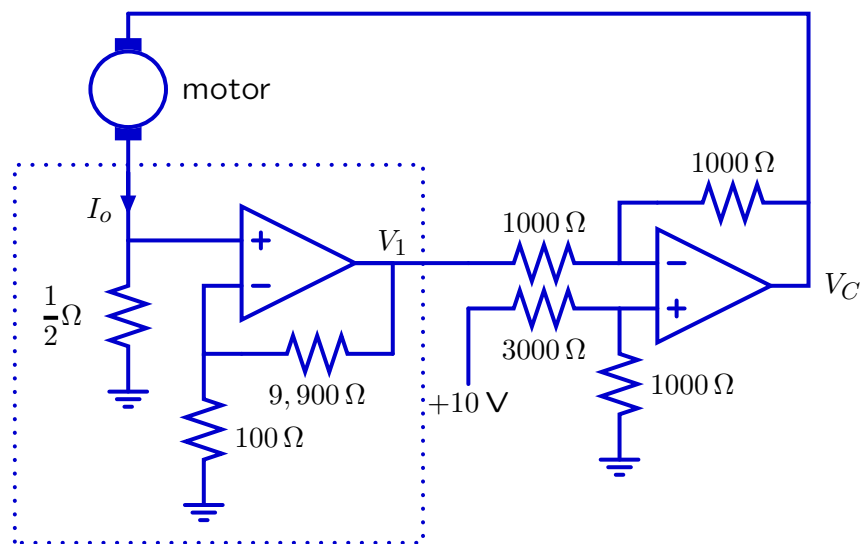
    startState = (0,0,0)
    def getNextValues(self, state, inp):
        (x1,x2,x3) = state
        y0 = inp - x3
        return ((inp,x1,x2), y0)
```

4 Motor Control (20 points)

The following circuit is a proportional controller that regulates the current through a motor by setting the motor voltage V_C to

$$V_C = K(I_d - I_o)$$

where K is the gain (notice that its dimensions are ohms), I_d is the desired motor current, and I_o is the actual current through the motor. Although K and I_d are not explicit in this circuit, their values can be determined from the resistor values below (see part b).



Part a. Consider the circuit inside the dotted rectangle. Determine an expression for V_1 as a function of I_o .

$$V_1 = \frac{1}{2} \Omega \times I_o \times 100$$

Part b. Determine the gain K and desired motor current I_d .

$K =$

$$50 \Omega$$

$I_d =$

$$0.1 \text{ A}$$

KCL at negative input to right op-amp:

$$\frac{V_C - 2.5 \text{ V}}{1000 \Omega} = \frac{2.5 \text{ V} - \frac{1}{2} \Omega \times I_o \times 100}{1000 \Omega}$$

Solving:

$$V_C - 2.5 \text{ V} = 2.5 \text{ V} - 50 \Omega \times I_o$$

$$V_C = 5 \text{ V} - 50 \Omega \times I_o = 50 \Omega (0.1 \text{ A} - I_o)$$

5 Members of the Club (20 points)

We would like to make a class to represent a club. A club has a list of members and a scoring function, which takes a member as an input and returns a numerical value. When a member is proposed for addition to the club, it is only added if its score is **greater than the average score of the current members**.

5.1 Join the club

We would like to make a club of basketball players, who are scored on their height. Here is a simple `BallPlayer` class.

```
class BallPlayer:
    def __init__(self, name, height):
        self.name = name
        self.height = height

    def getHeight(self):
        return self.height

    def __str__(self):
        return 'BallPlayer('+self.name+', ' + str(self.height) +')
```

The `Club` class has an `__init__` method that takes two arguments: an initial member, and a scoring function (that takes a single member as input and returns a numerical value).

Write an expression below that will create a new club. The first member is person named 'Wilt' whose height is 84 inches. The scoring function for club members should return their height.

```
c = Club(BallPlayer('Wilt', 84), BallPlayer.getHeight)
```

Now, imagine that we try, successively, to add the following new players, whose names and heights are listed below, to club `c`:

- 'Shorty', 60
- 'Walt', 86

- 'Stilt', 90
- 'Larry', 85

List the resulting membership of club c.

```
'Wilt', 'Walt', 'Stilt'
```

5.2 Implementation

Fill in the definition of the Club class. Use a list comprehension in the averageScore method.

```
class Club:
    def __init__(self, firstMember, scoreFunction):
        self.members = [firstMember]
        self.scoreFunction = scoreFunction

    # Returns average score of current members.
    def averageScore(self):

        n = float(len(self.members))
        return sum([self.scoreFunction(m) for m in self.members])/n

    # Adds member if it meets the criterion. Returns True if the
    # member was added and False, if not.
    def proposeMember(self, member):

        if self.scoreFunction(member) > self.averageScore():
            self.members.append(member)
            return True
        else:
            return False
```

5.3 Histogram

Write a procedure to compute a histogram of the member scores, that is, a count of how many members' scores fall within some specified ranges. We specify ranges by a list of N upper bounds. For example, the following bound list ($N = 3$):

```
[3, 9, 12]
```

specifies the following $N + 1 = 4$ ranges:

```
x < 3, 3 <= x < 9, 9 <= x < 12, 12 <= x
```

Given a list of scores that is already sorted from smallest to largest, such as: [1, 1, 4, 5, 7, 15] the resulting histogram would be: [2, 3, 0, 1] That is, 2 scores less than 3, 3 scores between 3 and 9, 0 scores between 9 and 12 and 1 score above 12.

```
>>> histogram([1, 1, 4, 5, 7, 15], [3, 9, 12])
[2, 3, 0, 1]
```

The output should always have $N + 1$ values; values should be zero if there are no scores in the appropriate range.

```
def histogram(scores, bounds):

    counts = []                # answer list
    index = 0                  # index into scores
    n = len(scores)           # total no of scores
    for x in bounds:          # for each bound
        count = 0              # initialize count
        while index < n and scores[index] < x:
            count += 1         # increment count
            index += 1         # move to next score
        counts.append(count)   # store count in answer
    counts.append(n-index)     # count any remaning scores
    return counts
```