Name:	**Cohort:**

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN
Established in collaboration with MIT

10.009 The Digital World
2013 Term 3

Date: 2 May 2013
Time: 2:00 p.m.
Duration: 2.5 hours

Instructions to candidates:

1. Write your name and cohort number at the top of this page.
2. This paper consists of 8 questions and 17 printed pages.
3. For questions 1–3, you are required to enter your answers into Tutor. For questions 4–8, you are required to write your answers in this exam booklet using a blue or black pen. All answers will be manually graded.
4. You may refer to the Digital World Notes and your personal notes.
5. You may refer to your copies of lab handouts and your solutions to lab questions.
6. You may write and test your code in IDLE.
7. You may use a web browser to refer to your personal copy of the libdw documentation, and your solutions to Tutor questions. No other Internet access allowed.
8. You may **not** communicate via any means with anyone (aside from the Digital World instructors).

For staff's use:

Qs 4	/12
Qs 5	/11
Qs 6	/9
Qs 7	/6
Qs 8	/12
Qs 4–8 Total	**/50**

1 Python: Tic-Tac-Toe *(26 points)*

Tic-tac-toe (naughts and crosses) is a game for two players, who take turns to mark `x`'s and `o`'s in a 3-by-3 grid. The player who places three marks in a row, column or diagonal first wins the game. We can show a tic-tac-toe grid on the screen by using `'_'` to represent an empty entry, and `'x'` and `'o'` to represent the two players, respectively. An example game can be:

1. Initialize the grid (`init`())

```
___
___
___
```

2. `x` moves (`movex`(b, 1, 3))

```
__x
___
___
```

3. `o` moves (`moveo`(b, 1, 1))

```
o_x
___
___
```

4. `x` moves (`movex`(b, 3, 1))

```
o_x
___
x__
```

5. `o` moves (`moveo`(b, 2, 2))

```
o_x
_o_
x__
```

6. `x` moves (`movex`(b, 3, 3))

```
o_x
_o_
x_x
```

7. `o` moves (`moveo`(b, 2, 3))

```
o_x
_oo
x_x
```

8. `x` moves (`movex`(b, 3, 2))

```
o_x
_oo
xxx
```

(Note that there is no space between adjacent underscores `'_'`.) By the end of step 8, the player `'x'` wins. If none of the players win after the grid is filled, the game is a draw. Other examples of winning situations are:

```
xxx    xo_    xo_
oo_    xo_    ox_
___    x__    __x
```

We use a list of lists to model the grid, where each sublist represents a row. For example,

```
__x
___
_o_
```

corresponds to the list

```
[['_', '_', 'x'],
 ['_', '_', '_'],
 ['_', 'o', '_']]
```

You are to write the following functions for this game:

1. `init()`, which takes no argument and returns a new grid *(2 points)*,

   ```
   >>> b = init()
   ```

```
>>> b
[['_', '_', '_'], ['_', '_', '_'], ['_', '_', '_']]
```

2. `show(b)`, which takes a grid as the only input argument, and returns a `str` that represents the grid. The returned string can be shown on the screen in exactly the same form as the examples above, where it consists of three lines, each line with three entries. There is a newline after each line, including the last line. *(4 points)*

3. `movex(b, i, j)` and `moveo(b, i, j)`, which take an input grid `b` and two input coordinates `i` and `j`, and place an `'x'` and an `'o'` into the grid at entry `(i,j)`, respectively. The functions do not return anything. Note that the coordinates start from 1. *(4 points)*

4. `countmoves(b)`, which takes an input grid as the only argument, and returns the number of moves that has been taken in the game. In the example game shown above, step 2 contains 1 move and step 8 contains 7 moves. *(4 points)*

5. `getmoves(b)`, which takes an input grid as the only argument, and returns a dict that contains all the moves of `'x'` and `'o'`, respectively. For example, if the input grid is:

   ```
   [['x', 'o', '_'],
    ['_', 'o', 'x'],
    ['_', '_', 'x']]
   ```

   the output should be:

   ```
   {'x': [(1, 1), (2, 3), (3, 3)], 'o': [(1, 2), (2, 2)]}
   ```

   Each list contains all moves of the corresponding player in order of their coordinates. *(6 points)*

6. `winsx(b)` and `winso(b)`, which return `True` if the corresponding player wins. The function should return `False` if the corresponding player does not win. *(6 points)*

Write your code and test it in IDLE, and **submit your answer via Tutor.**

2 Python: Visitor Class *(14 points)*

Write a class, `Visitor`, which records the number of visitors and the number of times each visitor calls on your cohort. A visitor can leave her/his name, or remain anonymous. In the latter case, we use `'Visitor` *n*`'` to represent the *n*th visitor to call on the cohort.

`Visitor` provides the following methods:

1. constructor, which takes an optional argument `name`, for the name of the `Visitor` object. (*5 points*)
2. `setName(name)`, which sets the name of a visitor to the value provided in the argument. *(3 points)*
3. `__call__`, which returns a string `'`*name* `called the` *m*`th time.'`, where *name* is the name of the visitor and *m* is the number of times the visitor object has called. For simplicity we use `th` even for the first and second visitors. *(6 points)*

Test case:

```
>>>v1=Visitor('John')
>>>print v1()
John called the 1th time.
>>>v2=Visitor()
>>>print v2()
Visitor 2 called the 1th time.
>>>print v1()
John called the 2th time.
>>>v2.setName('Marina')
>>>print v2()
Marina called the 2th time.
>>>v3=Visitor()
>>>print v3()
Visitor 3 called the 1th time.
```

Write your code and test it in IDLE, and **submit your answer via Tutor.**

3 SM: Fruitopia *(10 points)*

You need to design a state machine that describes the movement of an herbivore on an unbounded fixed grid. Since this herbivore eats only fruits, your state machine should take fruits as input. Different fruits cause different behaviors of the herbivore: the herbivore moves forward one unit after eating an apple, it turns 90 degrees clockwise (negative 90 degrees) after eating a pear, and 90 degrees counterclockwise (positive 90 degrees) after eating a plum. The herbivore starts at the position `(0, 0)` facing along the positive `x` axis. After eating each fruit, the herbivore outputs its current location.

Here is the `Fruit` superclass that defines a generic method.

```
class Fruit:
    def relativeMovement(self):
        return (self.relativeRotation, self.unitsForward)
```

Below are three subclasses `Apple`, `Pear`, and `Plum` of the `Fruit` class.

```
class Apple(Fruit):
    relativeRotation = 0
    unitsForward = 1

class Pear(Fruit):
    relativeRotation = -90
    unitsForward = 0

class Plum(Fruit):
    relativeRotation = 90
    unitsForward = 0
```

Each subclass encodes the displacement for each type of fruit. Using these subclasses, the following sorts of behavior are observed:

```
>>> a = Apple()
>>> a.relativeMovement()
(0, 1)

>>> p = Pear()
>>>p.relatveMovement()
(-90, 0)
```

Define an `Herbivore` class that is a subclass of `sm.SM` and implements the behavior of an herbivore. Inputs to the state machine are instances of the Fruit class. The output is a pair of indices indicating the herbivore's position on the grid. The initial location is `(x, y) = (0, 0)` and the herbivore starts out facing along the `x` axis (so that if it immediately moves forward one step it will be at `(1, 0)`). Below is an example using the `Herbivore` class.

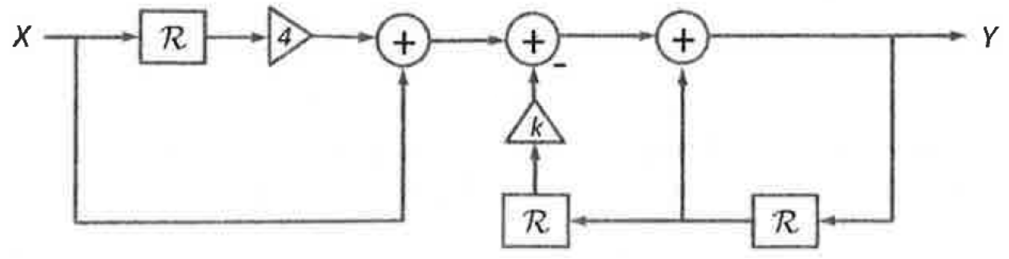
```
>>> h = Herbivore()
>>> fruits = [Apple(), Apple(), Pear(), Apple(), Pear(), Apple()]
>>> h.transduce(fruits)
[(1,0), (2,0), (2,0), (2,-1), (2,-1), (1,-1)]
```

Write your code and test it in IDLE, and **submit your answer via Tutor.**

4 Systems *(12 points)*

4.1 System Function *(4 points)*

What is the system function corresponding to this block diagram?



Write your answer and its derivation below:

4.2 Poles *(4 points)*

Compute the poles of the system $\frac{Y}{X} = \frac{1-3\mathcal{R}}{1+\mathcal{R}-kR^2}$ in terms of k in the box below. Show how you arrive at your answer.

4.3 Picking k *(4 points)*

Consider a system with poles $\frac{1 \pm \sqrt{1+3k}}{3}$.

a. For what range of real values of k (if any) is the output diverging (but not oscillating) in the long term?

b. For what range of real values of k (if any) is the output monotonically converging in the long term?

c. For what range of real values of k (if any) is the output oscillating and converging in the long term?

d. For what range of real values of k (if any) is the output oscillating and diverging in the long term?

5 System Representations *(11 points)*

Below is the definition for a state machine:

```
class mySM(sm.SM):
    startState = (0, 0, 0)
    def getNextValues(self,state,inp):
        output = 2 * state[2] + state[1] - 3 * state[0]
        newState = (output,state[0], inp)
        return (newState, output)
```

5.1 Difference Equations *(4 points)*

In the box below, write the difference equation that describes the same system as the state machine above.

5.2 Block Diagram *(3 points)*

In the box below, draw a block diagram that represents the same system as the state machine above.

5.3 System Function Code *(4 points)*

Define a procedure `mySF` that takes no arguments and returns an instance of `sf.SystemFunction` that represents the system shown in the state machine above. Do **NOT** directly create a new instance of `sf.SystemFunction`; rather, use the primitives and combinators defined in the `sf` module (e.g., `sf.R`, `sf.Gain`, `sf.Cascade`, `sf.FeedbackAdd`, `sf.FeedbackSubtract`, etc.) to construct this instance. **You do NOT need to enter your answer into Tutor.**

6 Signals *(9 points)*

Sketch the following signals. You may need different scales on the Y axis for different plots; please label the scale you use. Remember that signals consist of sample values at discrete time points. You do not have to calculate the numerical values to high precision. A graph showing roughly the right shape will be considered correct.

A. The response of a system with system function

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

to the signal $(1 + 4\mathcal{R}^4)\Delta$ (where Δ is the unit sample signal).

B. The unit-sample response of a system with system function.

$$\frac{Y}{X} = \frac{1}{1 + \frac{81}{100}\mathcal{R}^2}$$

C. The unit-sample response of a system with system function

$$\frac{Y}{X} = \frac{10}{1 + 0.5\mathcal{R}} + \frac{1}{1 - 0.99999999\mathcal{R}}$$

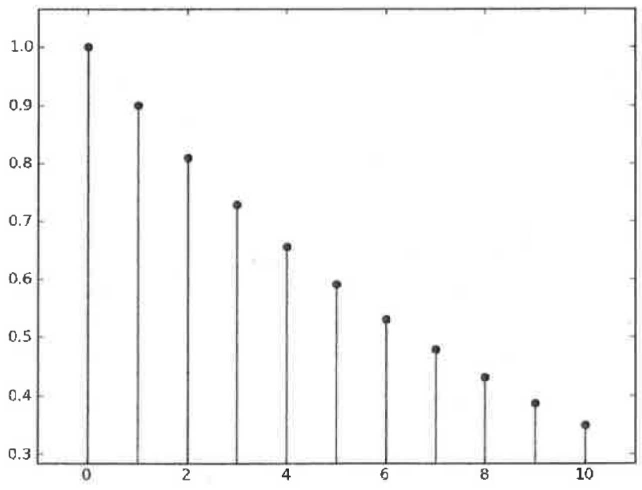
7 Making Signals *(6 points)*

Let H_1 to H_6 represent the following system functions:

- $H_1 = \frac{Y}{X} = \frac{P(\mathcal{R})}{1}$
- $H_2 = \frac{Y}{X} = \frac{1}{1+C\mathcal{R}}$
- $H_3 = \frac{Y}{X} = \frac{1}{(1+C\mathcal{R}^2)(1-C\mathcal{R}^2)}$
- $H_4 = \frac{Y}{X} = \frac{1}{1+C\mathcal{R}^2}$
- $H_5 = \frac{Y}{X} = \frac{C}{1-0.99999\mathcal{R}}$
- $H_6 = \frac{Y}{X} = \frac{1}{1+C\mathcal{R}} + \frac{D}{1-0.99999\mathcal{R}}$

where C and D represent unknown constants and $P(\mathcal{R})$ represents an unknown polynomial in $\mathcal{R}$. Each of the following plots represents the response of a linear, time-invariant system that can be represented by one of H_1 through H_6. The plots show just the first 11 samples of the response. You can assume that the pattern established for $0 \le n \le 10$ persists for $n > 10$. All responses are zero for $n < 0$. For each plot, determine the corresponding system function as well as the unknown constants and/or polynomial for that system function. You may choose a system function more than once or not at all.

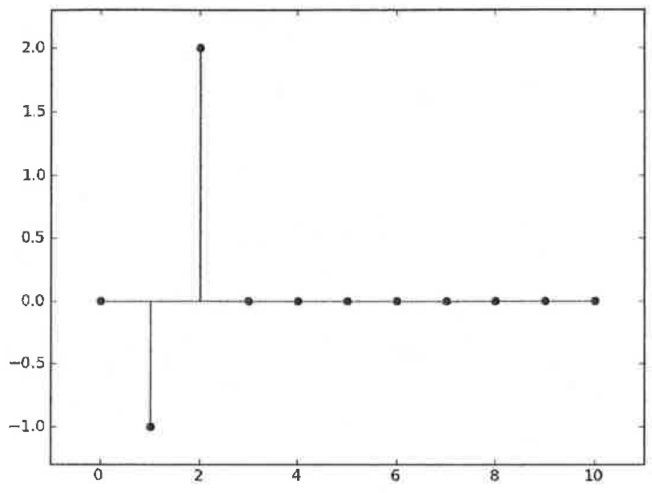
Part A. Specify the system function (H_1 to H_6) and corresponding constants and/or polynomial $P(\mathcal{R})$ that best describe the system whose unit sample response is shown below. The first three sample values are 1.0, 0.9, 0.81.



System Function:

C, D and/or $P(\mathcal{R})$:

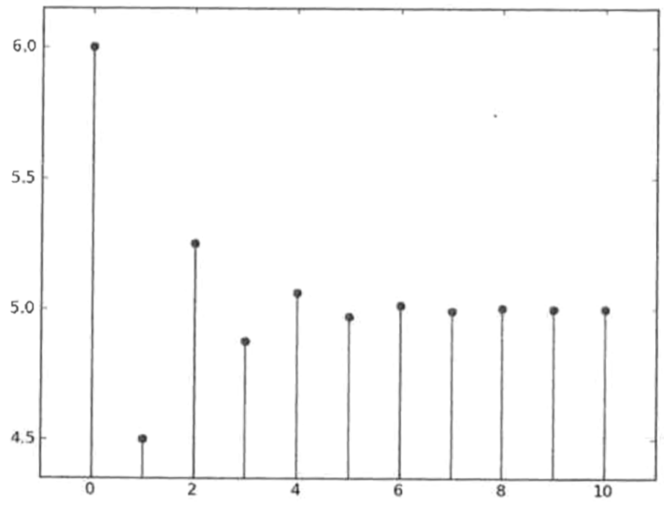
Part B. Specify the system function (H_1 to H_6) and corresponding constants and/or polynomial $P(\mathcal{R})$ that best describe the system whose unit sample response is shown below. The first three sample values are 0.0, -1.0, 2.0.



System Function:

C, D and/or $P(\mathcal{R})$:

Part C. Specify the system function (H_1 to H_6) and corresponding constants and/or polynomial $P(\mathcal{R})$ that best describe the system whose unit sample response is shown below. The first three samples are 6.0, 4.5, 5.25.



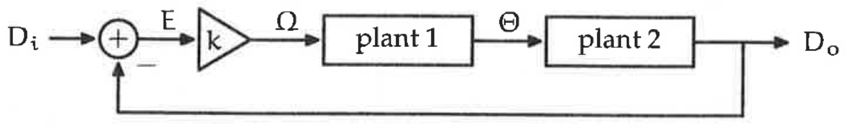
System Function:

C, D and/or $P(\mathcal{R})$:

8 Wall Following Labs *(12 points)*

In the "Staggering Proportions" and "Sizeable Following" labs, we investigated three different control strategies for keeping the robot at a constant, desired distance from a wall as the robot moved parallel to that wall. Block diagrams and associated difference equations are shown below for each of these control strategies, where $T = 0.1$ seconds and $V = 0.1$ m/s.

Proportional Model

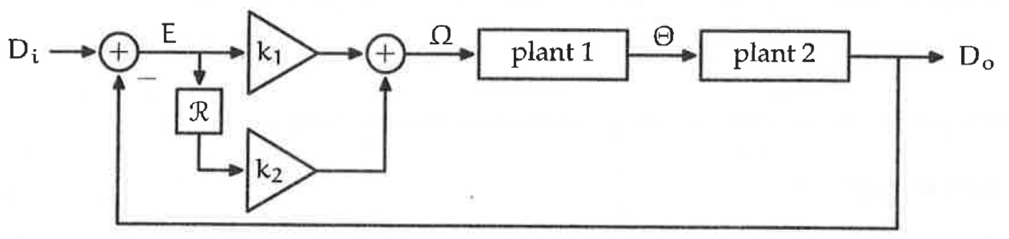


$$\omega[n] = ke[n]$$

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

$$d_0[n] = d_0[n-1] + VT\theta[n-1]$$

Delay-Plus-Proportional Model

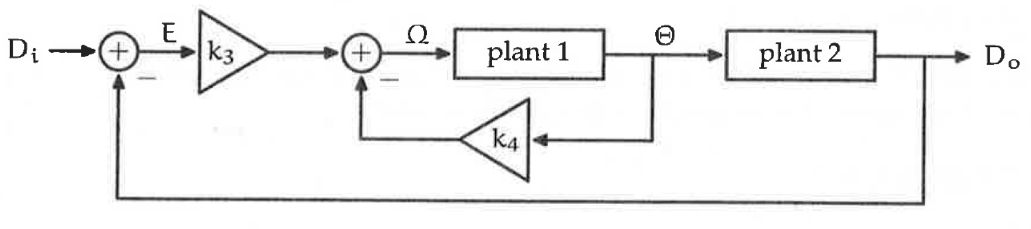


$$\omega[n] = k_1 e[n] + k_2 e[n-1]$$

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

$$d_0[n] = d_0[n-1] + VT\theta[n-1]$$

Angle-Plus-Proportional Model



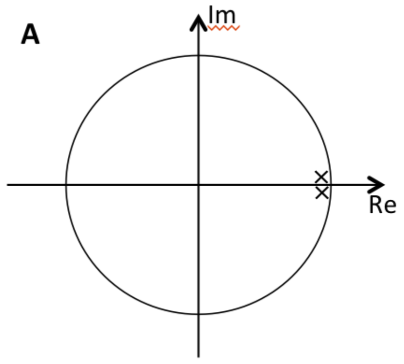
$$\omega[n] = k_3 e[n] - k_4\theta[n]$$

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

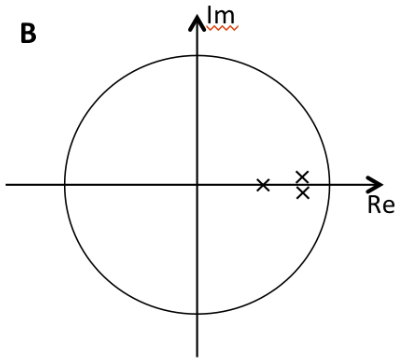
$$d_0[n] = d_0[n-1] + VT\theta[n-1]$$

Each of the following plots show the poles that result for one of the control strategies described below. The circles have unit radius.

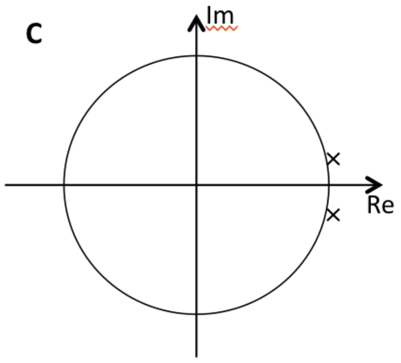
A



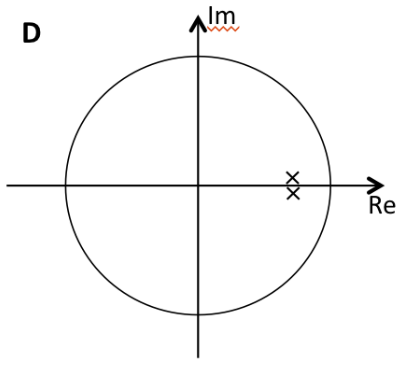
B



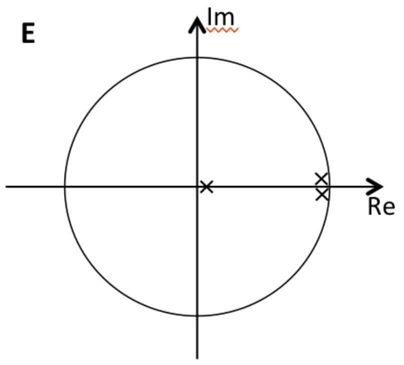
C



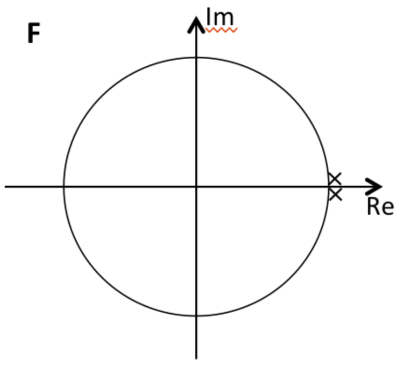
D



E



F



1. Which plot best describes the poles for the proportional model when $k = 10$?

A, B, C, D, E, or F:

2. Which plot best describes the poles for the proportional model when $k = 40$?

A, B, C, D, E, or F:

3. Which plot best describes the poles for the delay-plus-proportional model when $k_1 = 30$, and k_2 is chosen to minimize the magnitude of the dominant pole?

A, B, C, D, E, or F:

4. Which plot best describes the poles for the delay-plus-proportional model when $k_1 = 300$, and k_2 is chosen to minimize the magnitude of the dominant pole?

A, B, C, D, E, or F:

5. Which plot best describes the poles for the angle-plus-proportional model when $k_3 = 3$, and k_4 is chosen to minimize the magnitude of the dominant pole?

A, B, C, D, E, or F:

6. Which plot best describes the poles for the angle-plus-proportional model when $k_3 = 30$, and k_4 is chosen to minimize the magnitude of the dominant pole?

A, B, C, D, E, or F:

END OF PAPER