Name:

Student ID:

10.009 The Digital World

Term 3. January 26-May 1, 2015.

Final Exam

29 Apr 2015 (Wed): 9:15-11:15am. Room: Cohort Classroom

Most recent update: Apr 23, 2015

- Write your name and ID at the top of this page.
- This exam has two parts. You can secure a maximum of 100 points.
- For Part A (questions 1-2), and question 7(b), write your answers on this exam paper.
- For Part B (questions 3-6), and question 7(a), submit your solution to Tutor.
- You may consult all material on your laptop and in your notes. You may also use any book(s) as a reference.
- **You are not allowed to use any Internet accessing or communicating device during the exam.**
- **You are not allowed to consult anyone inside or outside of the classroom other than the proctors in the examination room.**
- Use Idle / Canopy to test your programs. Once you are satisfied, enter your program into Tutor and either save it or submit. In case you decide to save, then you MUST submit it before the end of the exam.
- All answers will be graded manually. You may be able to earn partial credit for questions.
- Good luck!

Summary:

Description	**Problems**	**Total Points**
Written questions	1, 2, 7(b)	28
Programming questions	3-6, 7(a)	72

1	
2	
7(b)	
SubTotal	

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Established in collaboration with MIT

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN HONOUR CODE

"As a member of the SUTD community, I pledge to always uphold honourable conduct. I will be accountable for my words and actions, and be respectful to those around me."

Introduction to the SUTD Honour Code

What is the SUTD Honour Code?

The SUTD Honour Code was established in conjunction with the school's values and beliefs, in good faith that students are able to discern right from wrong, and to uphold honourable conduct. It is an agreement of trust between the students and the staff and faculty of SUTD, and serves as a moral compass for students to align themselves to. Being in a university that aspires to nurture the leaders of tomorrow, it calls for students to behave honourably, not just solely in their academic endeavours, but also in everyday life.

What the Honour Code encompasses

Integrity & Accountability

To be honourable is to do what is right even when nobody is watching, and to be accountable for the things one does. One should always be accountable for one's words and actions, ensuring that they do not cause harm to others. Putting oneself in a favourable position at the expense of others is a compromise of integrity. We seek to create a community whereby we succeed together, and not at the expense of one another.

Respect

Part of being honourable is also respecting the beliefs and feelings of others, and to never demean or insult them. Should conflicts arise, the aim should always be to settle them in a manner that is non-confrontational, and try to reach a compromise. We will meet people of differing beliefs, backgrounds, opinions, and working styles. Understand that nobody is perfect, learn to accept others for who they are, and learn to appreciate diversity.

Community Responsibility

In addition to that, being honourable also involves showing care and concern for the community. Every individual has a duty to uphold honourable conduct, and to ensure that others in the community do likewise. The actions of others that display immoral or unethical conduct should not be condoned nor ignored. We should encourage each other to behave honourably, so as to build a community where we can trust one another to do what is right.

____

Student's signature

<u>Part A</u>

Q.1 [Total: 15 points]

This question is related to graphical user interfaces (GUI) programming using Tkinter.

a) [5 points] Explain what is a Widget in GUI programming.

b) [6 points] What is the mainloop() method in Tk() class for?

c) [4 points] Name two Geometry Managers in Tkinter, i.e., two different mechanisms to arrange widgets in a Tkinter GUI application.

<u>Your Answer for question 1:</u>

Q.2 [5 points]

In week-10, you have worked on a proportional controller to keep the robot a fixed distance from the wall in front of it. In particular, we discussed a proportional controller in which the robot's velocity is proportional to the difference between its desired distance to the wall and its actual distance to the wall. In the following code (for AmigoBot), the output of the `Sensor` state machine (which represents the sensed distance to the wall, `dSensed`) is the input to the `Controller` state machine.

In the `Controller` state machine, `k` represents the constant of proportionality between forward velocity and the difference (`dDesired-dSensed`).

Please suggest a value of `k` in the `Controller` state machine so that the sensed distance can converge to the desired distance. Specifically, what should be the sign of `k`? Explain your answer.

```
import math
import libdw.sm as sm
from soar.io import io
import libdw.gfx as gfx
import libdw.util as util

dDesired = 0.7
k = # Please suggest a value of k

# Input is output of Sensor machine (below); output is an action.
# Note that this machine must also compute E, the error, and output
# the velocity, based on that.
class Controller(sm.SM):
    def getNextValues(self, state, inp):
        fvel = k * (dDesired - inp)
        rvel = 0
        return state, io.Action(fvel, rvel)

# Input is SensorInput instance;
# output is a delayed front sonar reading
class Sensor(sm.SM):
    def __init__(self, initDist, numDelays):
        self.startState = [initDist]*numDelays
    def getNextValues(self, state, inp):
        print inp.sonars[3]
        output = state[-1]
        state = [inp.sonars[3]] + state[:-1]
        return (state, output)

mySM = sm.Cascade(Sensor(1.5, 1), Controller())
mySM.name = 'brainSM'
```

Your Answer for question 2:

Part B

Q.3 [10 points]

Write a function `compTrace(A)` to compute the trace of a matrix `A`. Specifically, given an `n`-by-`n` square matrix `A = [`a_{ij}`]`, with a_{ij} denotes the element on the `i`-th row and `j`-th column of `A`, the trace of `A` is defined to be the sum of the elements on the main diagonal (the diagonal from the upper left to the lower right) of `A`. That is, `compTrace(A)` returns:

$$a_{11} + a_{22} + \ldots + a_{nn}$$

The `n`-by-`n` matrix `A` is represented by Python nested lists. Each matrix is a nested list that has `n` entries, each entry is a list that represents a row of the matrix.

Sample tests:

Input:

```
A = [[2.2, 2, 3.1], [4, 5, 6], [7, 8, 9]]
```

Output:

```
16.2
```

Q.4 [10 points]

A dictionary `dInput` has integer key and string value. Write a function

```
findKey(dInput, strInput)
```

`findKey()` takes a dictionary `dInput` and a string `strInput` as inputs, and returns a list that contains all the keys which values equal to `strInput`. The returned list should contain the integer keys in ascending order. An empty list is returned if there is no key that has the value of `strInput`.

Sample tests:

```
dInput = {1:'singapore', 20:'china', 4:'japan', 5:'china', 10:'japan'}

findKey(dInput, 'china') returns [5, 20]

findKey(dInput, 'korea') returns []
```

Q.5 [Total: 30 points]

Design a class named `Square`. The class contains:

a) [1 point] Two `float` data fields named `x` and `y` that specify the center of the square (Assume that the square sides are parallel to x- or y- axes.)

b) [1 point] The `float` data field `sideLength` that specifies the length of one side of the square. You can assume `sideLength > 0`.

c) [4 points] A constructor that creates a square with the specified `x, y, sideLength`. The default value for `x` and `y` is `0`. The default value for `sideLength` is `1.0`.

d) [4 points] The accessor method `getCenter()` that returns the center of the square as a `tuple`.

e) [4 points] The accessor method `getSideLength()` that returns the value of the `sideLength` attribute.

f) [4 points] A method named `getArea()` that returns the area of the square.

g) [4 points] A method named `getPerimeter()` that returns the perimeter of the square.

h) [4 points] A method `containPoint(px, py)` that returns `True` if the specified point `(px, py)` is inside this square, returns `False` otherwise. See Figure 5(a). If the point `(px, py)` is on the boundary of the square, `containPoint(px,py)` also returns `True`.

i) [4 points] A method `containSquare(inSquare)` that returns `True` if the specified `Square inSquare` is *completely* inside this square. See Figures 5(b), 5(c), 5(d). Please see sample tests.

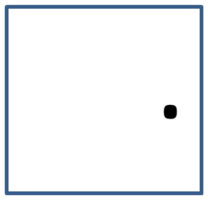

Figure 5(a)
`containPoint(px,py)`
returns `True`

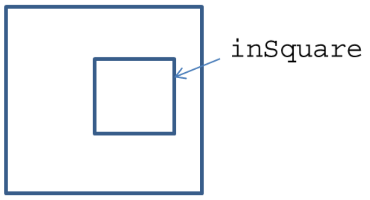

Figure 5(b)
`containSquare(inSquare)`
returns `True`

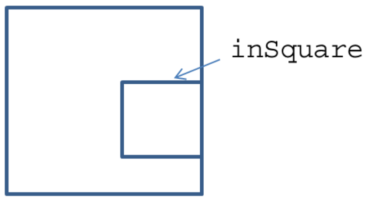

Figure 5(c)
`containSquare(inSquare)`
returns `True`

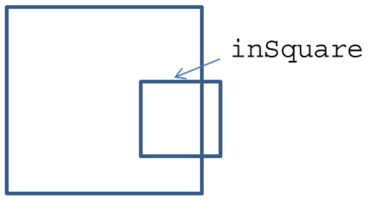

Figure 5(d)
`containSquare(inSquare)`
returns `False`

Sample tests:

```
>>> s = Square(x=1,y=1, sideLength=2.0)

>>> print s.getCenter()

(1, 1)

>>> print s.getSideLength()

2.0

>>> print s.getArea()

4.0

>>> print s.getPerimeter()

8.0

>>> print s.containPoint(0,0)

True

>>> print s.containPoint(0,-0.5)

False

>>> print s.containPoint(1,1.5)

True

>>> print s.containSquare( Square(x=1.5, y = 1, sideLength = 1))

True

>>> print s.containSquare( Square(x=1.5, y = 1, sideLength = 1.1))

False

>>> s2 = Square()

>>> print s2.getCenter()

(0, 0)

>>> print s2.getSideLength()

1.0

>>> print s2.getPerimeter()

4.0
```

Q.6 [15 points]

In this problem you will implement in Python the behavior of a simplified elevator. The behavior of this elevator can be captured as a finite state machine (FSM). The elevator can be at one of three floors: 'First', 'Second', 'Third'. There is one button that controls the elevator, and it has two values: 'Up' or 'Down'. The elevator has a display that shows the current floor.

At each time step, the elevator checks the current floor and the current value of the button, changes floor and display in the obvious way: The elevator moves up one floor if the current button value is 'Up'. When it reaches the 'Third' floor, it remains at the 'Third' floor if button value is 'Up'. Likewise, the elevator moves down one floor if the current button value is 'Down'. When it reaches the 'First' floor, it remains at the 'First' floor if button value is 'Down'. Please see sample interaction below.

Write a class named `Elevator` that contains attributes and methods as described below.

`Elevator` class is a subclass of `sm.SM` class, which is part of `libdw`.

The Elevator starts from the 'First' floor.

The Elevator class has a method named `getNextValues(self, state, inp)` that takes the current state and input and returns the next state and output. At each step, the input to the Elevator FSM is the button value ('Up' or 'Down'). The output of the Elevator FSM at each step is the floor display ('First', 'Second' or 'Third').

The button value can either be 'Up' or 'Down', but cannot be both at the same time.

Sample interaction:

```
e = Elevator()

print e.transduce( ['Up', 'Up', 'Up', 'Up', 'Down', 'Down', 'Down', 'Up'])
```

Output:

```
['Second', 'Third', 'Third', 'Third', 'Second', 'First', 'First', 'Second']
```

Q.7 [15 points]

Suppose you are in a hallway lined with `K` closed lockers. You begin by opening all `K` lockers. Next, you close every second locker. Then you go to every third locker and close it if it is open or open it if it is closed (called this toggling the lockers). You continue toggling every `n`th locker on pass number `n`. After your `K`-th pass of the hallway, in which you toggle only locker number `K`, how many lockers are open? You can assume `K > 0`.

a) [7 points] Write a function `countNumOpenLocker(K)` that takes the total number of lockers `K` as input and returns the number of lockers that are open at the end of the `K`-th pass.

b) [8 points] How many lockers are open at the end of the `K`-th pass, with `K=1,000,000`? Most likely, you would not be able to compute this with your program. Try to derive this analytically with logical thinking and problem solving skills. Explain your answer.

The following figure depicts the case when `K=6`. ‘c’ indicates that the locker is closed. ‘o’ indicates that the locker is open. In this case, two lockers are open at the end of the `K`-th pass. Thus, `countNumOpenLocker(6)` returns `2`.

	1st locker	2nd locker	3rd locker	4th locker	5th locker	6th locker
	c	**c**	**c**	**c**	**c**	**c**
1st pass	**o**	**o**	**o**	**o**	**o**	**o**
2nd pass	o	**c**	o	**c**	o	**c**
3rd pass	o	c	**c**	c	o	**o**
4th pass	o	c	c	**o**	o	o
5th pass	o	c	c	o	**c**	o
6th pass	o	c	c	o	c	**c**

Note: For 7(a), submit your solution to Tutor. For 7(b), write your answer and explanation on this exam paper. Full credit will be given to 7(b) only if the answer is adequately explained.

Sample tests:

```
countNumOpenLocker(2000) returns 44

countNumOpenLocker(10) returns 3

countNumOpenLocker(20) returns 4
```

Your Answer for question 7(b):

This Page Intentionally Left Blank. For Your Own Use.

End of Exam Paper