Name:

Student ID:

10.009 The Digital World

Term 3. 2017.

Final Exam

April 28, 2017 (Friday)

Overall Timing: 3:00-5:30pm. Actual Duration: 3:15pm- 5:15pm (2 hours)

Room: Cohort Classroom

Most recent update: April 28, 2017

- Write your name and student 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), write your answers on this exam paper.
- For Part B (questions 3-7), submit your solutions to Tutor. Template files may be provided in 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 your desktop IDE 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:

Category	Description	Number of Problems	Total Points
Part A	Written questions	2 (Questions 1-2)	20
Part B	Programming questions	6 (Questions 3-7)	80

Q1	
Q2	
SubTotal	

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

[this page is left blank]

Part A

Q.1 [10 points]

One student was not satisfied with his boundary follower program for the mini project. He wanted to rotate exactly 90 degrees when the eBot found a boundary. To do that, he wrote a state machine to test his 90 degrees rotation as shown below. The eBot should move forward until it sees a boundary at around 0.5 m and starts rotating. After it rotates for 90 degrees, it should move forward again.

```
1   class MySMClass(sm.SM):
2       startState=('forward',0.0)
3       def getNextValues(self, state, inp):
4           state, orig_angle = state
5           angle = util.fixAnglePlusMinusPi(inp.odometry.theta)
6           front_dist = inp.sonars[2]
7           eps = 0.01
8           if state=='forward':
9               if front_dist <=0.5:
10                  next_state = ('rotate', angle)
11                  forward = 0.0
12                  rotation = 0.1
13              else:
14                  next_state = (Q1, Q2)
15                  forward = 0.1
16                  rotation = 0.0
17          elif state == 'rotate':
18              if not util.nearAngle(abs(angle-orig_angle), math.pi/2.0,eps):
19                  next_state = (Q3, Q4)
20                  forward = 0.0
21                  rotation = 0.1
22              else:
23                  next_state = (Q5, Q6)
24                  forward = 0.1
25                  rotation = 0.0
26          print next_state, forward, rotation
27          return (next_state, io.Action(fvel = forward, rvel = rotation))
```

The code above can be downloaded from: http://10.1.3.25/q1_code.py .

a) [3 points] Specify the values of Q1 to Q6 at lines 14, 19, and 23.

b) [5 points] Draw the state transition diagram when you consider only the first item in the tuple of your state variable. Hint: you can use $\Delta\theta$ to denote **`abs(angle - orig_angle)`** in your state diagram.

c) [1 points] Explain the purpose of using **`util.fixAnglePlusMinusPi()`**. What problems may be encountered if we do not use this function? Hint: Check **`libdw.util`** module documentation.

d) [1 point] Draw one kind of world with its boundary where this state machine will not work.

Your Answer:

Q.2 [10 points]

My friend Julius-C explained to me this awesome trick to share secret messages. He told me it was super secure back in the day. You just shift each letter of your message according to its order in the alphabet, by a super secret constant factor! So for instance when ciphering "AB" with key=3, the result is "DE". To decrypt, one just shifts back the same number of positions. I realized this was easy to implement in Python with my DW skills. To test my implementation he sent me an encrypted message for me to decrypt. But something went wrong with my code below :(

```
1   alphabet = {'A':0,'B':1,'C':2,'D':3,'E':4,'F':5,'G':6,'H':7,'I':8,
2   'J':9,'K':10,'L':11,'M':12,'N':13,'O':14,'P':15,'Q':16,'R':17,'S':18,
3   'T':19,'U':20,'V':21,'W':22,'X':23,'Y':24,'Z':25}
4   # the next line is just to create the inverse alphabet dictionary
5   inverse = {v: k for k, v in alphabet.items()}
6
7   #Encrypting function
8   def cipher(m,key):
9       c = ""
10      for x in m:
11          y = alphabet[x] + key  % 26
12          c = c + inverse[y]
13      return c
14
15  #Decrypting function
16  def decipher(c,key):
17      m = ""
18      for y in c:
19          x = alphabet[y] - key  % 26
20          m = m + inverse[x]
21      return m
22  #I'm not sure if this was the key, I forgot!
23  key = 5
24  #But I'm pretty sure that the secret messages starts with 'D'..
25  c = "QJEHYRM"
26  #print cipher(m,k)
27  print decipher(c,key)
```

The code above can be downloaded from: http://10.1.3.25/q2_code.py .

a)[5 points] Can you spot and fix the logical error in the implementation? This is the run-time error I get when running the code:

```
line 20, in decipher
    m = m + inverse[x]
KeyError: -1
```

Note: There are no syntax errors in the code, I checked!

b)[5 points] I thought the key was 5 but I think I forgot. Can you help me recover the secret key and the secret message? Julius-C told me that the first letter of the secret message was "D". Explain your answer.

Your Answer:

Part B

Q.3 [10 points]

All books published before 2007 were assigned a 10-digit reference number called ISBN (International Standard Book Number). The 10 digits can be denoted as: $d_1d_2d_3d_4d_5d_6d_7d_8d_9d_{10}$. The last digit, d_{10}, is called 'checksum', which is calculated from the other nine digits using the following formula:

$$(d_1 \times 1 + d_2 \times 2 + d_3 \times 3 + d_4 \times 4 + d_5 \times 5 + d_6 \times 6 + d_7 \times 7 + d_8 \times 8 + d_9 \times 9) \% 11$$

If the checksum is 10, the last digit is denoted as 'X', according to the ISBN convention. Write a function named **`complete_ISBN`** that takes the first 9 digits as one string argument, then returns the full 10-digit ISBN number (including leading zeros if any) as a string.

Sample tests:	**Output:**
`print "Test case 1: input=013601267"` `print complete_ISBN("013601267")`	`0136012671`
`print "Test case 2: input=013031997"` `print complete_ISBN("013031997")`	`013031997X`
`print "Test case 3: input=020139829"` `print complete_ISBN("020139829")`	`020139829X`

Q.4 [15 points]

Write a function **`get_products(inlist, test)`**. The function returns two outputs (as a tuple). The first output is a dictionary **`d`**. For each tuple in **`inlist`**, the product of the entries is calculated. The products calculated are the keys of the dictionary **`d`**. The corresponding values is a list of the tuples that give the product. For example,

```
inlist = [(2,2), (1,4), (2,3) ]
gives d = {4: [(2,2),(1,4)],6: [(2,3)]}
```

The second output is a list of tuples from **`inlist`** that have **`test`** as the product of its entries. If there is no corresponding value, the second output should be a **`None`** object.

Sample input list:

```
inlist = [(3,5),(2,2),(2,2,3),(12,2),(7,3),(3,7,1)]
```

Test Code:	**Output:**

`d,o = get_products(inlist, 15)`
`print sorted(d.keys())`
`print sorted(d.values())`
 	
`print o`
`d,o = get_products(inlist, 21)`
`print o`
`d,o = get_products(inlist, 11)`
`print o`
	`[4,12,15,21,24]`
	`[[(2,2)], [(2,2,3)], [(3,5)], [(7,3), (3,7,1)], [(12,2)]]`
	`[(3,5)]`
	`[(7,3),(3,7,1)]`
	`None`

Q.5 [15 points]

A fictional language for an upcoming fantasy television show has been invented using the letters of the roman alphabet. Soon, the scriptwriters will have lots of text written for this language, and your task is to build a spell checker program for documents produced in this language.

The rules of the written form of this language is as follows.

- Each word must have only two letters.
- The first letter must be one of the following lower-case consonant letters: k g s t d n h b m r
- The second letter must be one of the following lower-case vowel letters: a e i o u
- There must be at least one space after the end of each word.

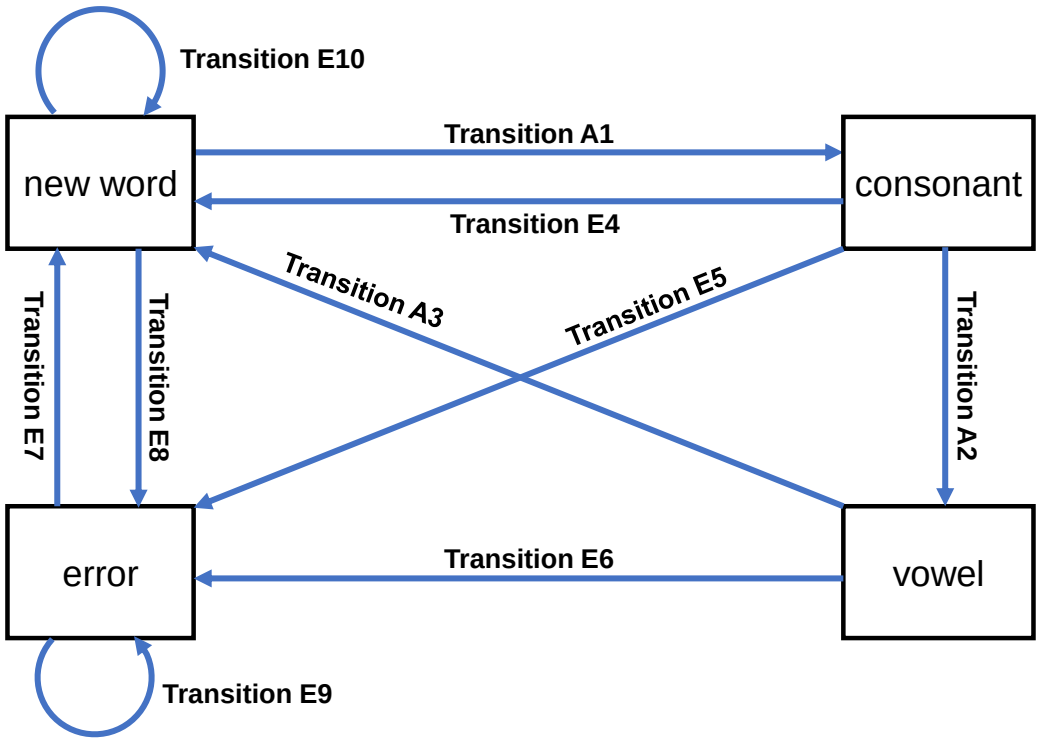
This language does not have upper-case letters or punctuation.

This spell-checker can be implemented using a finite state machine. Write a class named `SpellCheckSM` to implement this spell-checking state machine. `SpellCheckSM` is a subclass of `sm.SM`, which is obtained by having `from libdw import sm` at the start of your code.

There are four states, to be named in your program as:

(1) `'new word'` (2) `'consonant'` (3) `'vowel'` (4) `'error'`

The state transition diagram is shown below. The various transitions are shown in the following page.



Transition	The input character:	The output:
A1	is a valid consonant	Empty string
A2	is a valid vowel	Empty string
A3	is a blank space	`'ok'`
E4	is a blank space	`'error'`
E5	is not a valid vowel AND is not a blank space	Empty string
E6	is not a blank space	Empty string
E7	is a blank space	`'error'`
E8	is not a valid consonant AND is not a blank space	Empty string
E9	is not a blank space	Empty string
E10	is a blank space	Empty string

Decide what the starting state should be and implement these transitions in `SpellCheckSM`.

The following are the sample tests. Note that all inputs `line` end with a single blank space.

Sample Tests	Output

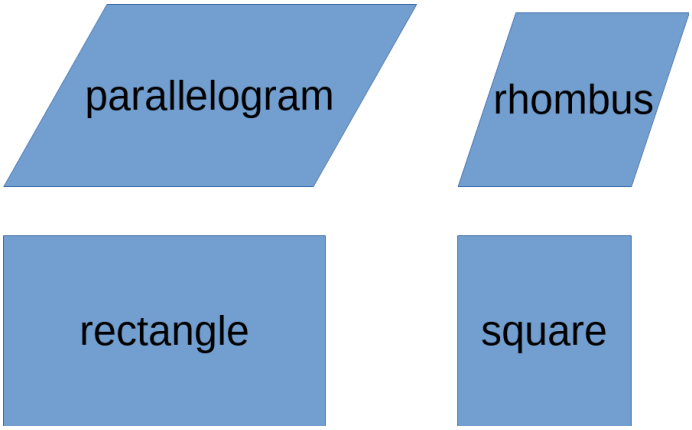
`print 'test case A'`
`a = SpellCheckSM()`
`line = 'a si tu ne mai me pas je '`
`print a.transduce(line)`
	`test case A`
	`['', 'error', '', '', 'ok', '', '',`
	`'ok', '', '', 'ok', '', '', '',`
	`'error', '', '', 'ok', '', '', '',`
	`'error', '', '', 'error']`

`print 'test case B'`
`a = SpellCheckSM()`
`line = 'hi ka ru no de '`
`print a.transduce(line)`
	`test case B`
	`['', '', 'ok', '', '', 'ok', '', '',`
	`'ok', '', '', 'ok', '', '', 'ok']`

`print 'test case C'`
`a = SpellCheckSM()`
`line = 'mu '`
`a.transduce(line,verbose=True)`
	`test case C`
	`Start state: new word`
	`In: m Out:  Next State: consonant`
	`In: u Out:  Next State: vowel`
	`In:  Out: ok Next State: new word`

Q.6 [Total: 25 points]

A parallelogram is a quadrilateral with two pairs of parallel sides. The opposite sides of a parallelogram are equal and the opposite angles of a parallelogram are equal. A rhombus is a parallelogram in which all four sides are equal. A rectangle is a parallelogram in which all angles are right angles. A square is a rectangle in which two adjacent sides have equal length.



The area of a square of side p is

$$Area = p^2$$

The area of a rectangle with sides a and b is

$$Area = a \times b$$

The area of a rhombus of each side x and height h is

$$Area = x \times h$$

The area of a parallelogram with two adjacent sides m and n, and a diagonal

d is

$$Area = 2\sqrt{s(s-m)(s-n)(s-d)}$$

Where $s = \frac{m+n+d}{2}$

Part 1

a) [5 points] Define a Parallelogram class which have the following *attributes*:

- side1 = One side of the parallelogram, as a float

- side2 = Another side of the parallelogram, adjacent to side1, as a float

and the following *property*:

- diagonal = A diagonal of the parallelogram, as a float

Implement the **`__init__()`** method of Parallelogram such that we can initialize an object of the class. For example, if we want to initialize side1 = 2, side2 = 3, and diagonal = 4:

```
para = Parallelogram(2,3,4)
```

Implement the **`__str__()`** method of the Parallelogram class such that it returns a string like "4.00" when the diagonal of the Parallelogram is 4. The number should be formatted to two decimal places. Your code should pass the following test cases.

Test case 1

Test code:	Output:
`para = Parallelogram(2,3,4)` `print para`	`4.00`

b) [5 points] Implement appropriate getter and setter methods for the diagonal property such that your code passes the following test case. Note that if the user tries to set a negative value for the diagonal, the setter method should set the value to 0.

Test Case 2

Test code:	Output:

`para = Parallelogram(2,3,4)`
`para.diagonal = 3`
`print para`
	`3.00`

Test Case 3

Test code:	Output:

`para = Parallelogram(2,3,4)`
`para.diagonal = -1`
`print para`
	`0.00`

Part 2

c) [10 points]

Define a Rhombus class that is a subclass of Parallelogram.

Define a Rectangle class that is a subclass of Parallelogram.

Define a Square class that is a subclass of Rectangle.

Implement the **`__call__`** method(s) such that for any of the shapes being initialized, it returns a Boolean True or False, on the basis of whether that particular shape is valid. Your code should pass the following test cases. **Note**: to be a parallelogram, side1, side2, and diagonal must form a triangle. The sum of the lengths of any two sides of a triangle is greater than the length of the third side.

Test case 4

Test code:	Output:
`para = Parallelogram(3,4,5)` `print para()`	`True`

Test case 5

Test code:	Output:
`para = Parallelogram(3,4,8)` `print para()`	`False`

Test case 6

Test code:	Output:
`rect = Rectangle(3,4,6)` `print rect()`	`False`

Test case 7

Test code:	Output:
`rhom = Rhombus(3,3,2)` `print rhom()`	`True`

Test case 8

Test code:	Output:

`from math import sqrt`
`squr = Square(2,2,3)`
`print squr()`
 	
`squr = Square(2,2,sqrt(8))`
`print squr()`
	`False`
	
	`True`

d) [5 points] Implement a **`calc_area()`** method such that for any of the four shapes defined above, the method returns the area of the shape as a float (rounded to 2 decimal places). Use your judgement to decide which class(es) to implement the **`calc_area()`** method in. Your code should pass the following test cases:

Test case 9

Test code:	Output:
`para = Parallelogram(3,4,2)` `print para.calc_area()`	`5.81`

Test case 10

Test code:	Output:
`para = Parallelogram(5,7,9)` `print para.calc_area()`	`34.82`

Test case 11

Test code:	Output:
`rect = Rectangle(3,4,5)` `print rect.calc_area()`	`12.0`

Test case 12

Test code:	Output:
`rhom = Rhombus(3,3,4)` `print rhom.calc_area()`	`8.94`

Test case 13

Test code:	Output:
`squr= Square(2,2,2.83)` `print squr.calc_area()`	`4.0`

Q.7 [15 points] Learn the art of procrastination.

If we can do something later, why do it now? Let us try to write a program to plan the procrastination. Given n tasks due today. The i-th task takes x_i units of time and must be finished by time t_i. Suppose we can only work on one task at a time, and once we begin a task, we must work until it is finished. What is the latest time that we must start to ensure that all the deadlines are met? Note that time units are integer and start from 0.

Write a function: **`procrastination(assignments)`**

The input "assignments" is a list of **`MyTask`** objects. **`MyTask`** has two attributes: int deadline, i.e., t_i, and int duration, i.e., x_i.

The code for **`MyTask`** is as follow (given in Tutor):

```
class MyTask(object):

      def __init__(self, deadline, duration):
            self.deadline = deadline
            self.duration = duration

      def __str__(self):
            return 'T(%d,%d)' %(self.deadline, self.duration)
```

For example:

assignments = [MyTask(9,1), MyTask(9,2), MyTask(7,1)]

Here "assignments" is a list of **`MyTask`** objects. The "assignments" has three tasks, the first one takes 1 unit of time, must be finished by time 9; the second one takes 2 unit of time, must be finished by time 9; the third one takes 1 unit of time, must be finished by time 7.

The function returns an integer indicating the latest time that we should start yet still manage to finish all the tasks on time. If the latest time would require us to start before time 0, return -1. You only need to finish the function **`procrastination()`**.

Test code:	Expected Output:

`assignments = [ MyTask(9,1), MyTask(9,2), MyTask(7,1) ]`
`print procrastination(assignments)`
`assignments1 = [ MyTask(3,2), MyTask(3,2) ]`
`print procrastination(assignments1)`
`assignments2 = [ MyTask(9,1), MyTask(9,2), MyTask(4,3) ]`
`print procrastination(assignments2)`
`assignments3 = [MyTask(14,10), MyTask(33,2), MyTask(5,3),`
`MyTask(14,1), MyTask(10,2)]`
`print procrastination(assignments3)`
	5
	-1
	1
	-1

End of Exam Paper

[this page is left blank]

[this page is left blank]