Name:

Student ID:

10.009 The Digital World

Term 3. 2017.

Midterm Exam

March 15, 2017 (Wednesday) 2:30-5:00pm. Room: Cohort Classroom

Most recent update: March 7, 2017

- Write your name and student ID at the top of this page.
- This exam has three parts. First try to complete Part A and Part B. 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.
- 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

Part A

Q.1 [10 points]

Consider the following code in a Python script file and answer the following questions.

```
 1 import math
 2 def combination(credit):
 3     credit = (credit - 10.0)/20.0
 4     out = 2.0*credit - 1.5
 5
 6     return out
 7
 8 def activation(a):
 9
10     probability = 1.0/(1 + math.exp(-a))
11     print probability
12
13 credit = 16.0
14 output = activation( combination(credit) )
15 print credit
16 print output
17
```

(a) A student calculates that when line 3 is executed, `credit` will have a value of 0.3. However, when line 15 is executed, he is surprised to see `16.0` displayed on the screen. Explain why this is the case. (4 points)

(b) When line 14 is executed, describe the sequence of the function calls and how data is passed among these functions and back to the variable `output`.
Hence, explain why when line 16 is executed, `None` is displayed on the screen. (6 points)

Your Answer:

Q.2 [10 points]

A student was asked to write a function to rotate the robot on itself a given number of times, either in clockwise or counterclockwise rotation. The student notes that it takes about 5 seconds to do a full rotation at 100% speed on one of the wheels. However, when testing his program, he realized something went wrong.

a) Can you spot any syntax error in the code (including type errors) that would prevent the program to run? Please indicate the lines and how would you correct them. (3 points)
b) After fixing syntax errors, can you spot any logical mistakes in the code? Please indicate the lines and how would you correct them. Briefly summarize the behavior of the robot for the given test case if the logical mistakes are not fixed. (7 points)

```
 1 from eBot import eBot
 2 import import time
 3
 4 # Rotate and conquer
 5 def one_round(speed):
 6     if direction = 'clockwise':
 7         ebot.wheels(0,1)
 8     elif direction = 'counter clockwise':
 9         ebot.wheels(1,0)
10     time.sleep(2.5)
11
12 def rotate(times, direction):
13     print "Rotating " + times + \
14           " times in direction: " + direction
15     for i in range(1,times):
16         one_round(speed)
17
18 ebot = eBot.eBot() # create an eBot object
19 ebot.connect() # connect to the eBot via Bluetooth
20
21 #Test case, this should rotate 3 full times clockwise...
22 rotate(3, clockwise)
23 #Prof! My code does not work!
24
25 ebot.disconnect() # disconnect the Bluetooth communication
```

Provide your answer on the next page.

Your Answer:

Part B

Q.3 [5 points]

A regular polygon is an n-sided polygon in which all sides are of the same length and all angles have the same degree. The formula for computing the area of a regular polygon is

$$Area = \frac{n \times s^2}{4 \times \tan\left(\frac{\pi}{n}\right)}$$

Here, s is the length of a side. Write a function named **`area_r_polygon`** that takes the number of sides and the length of a side as arguments, then returns the area of the regular polygon up to 3 decimal places. Note: Use **`math.pi`** to obtain an accurate value for pi.

Sample tests:	**Output:**
`print ''Test case 1: n=5, s=6.5''` `print area_r_polygon(5, 6.5)`	`72.69`
`print ''Test case 2: n=7, s=3.25''` `print area_r_polygon(7, 3.25)`	`38.383`
`print ''Test case 3: n=2, s=12.5''` `print area_r_polygon(2, 12.5)`	`0.0`

Q.4 [10 points]

The number 3, 5, 6, and 9 are all integers below 10 that are multiples of either 3 or 5; the sum of 3, 5, 6, 9 is 23. Similarly, 2, 4, 6, 8, 10 are all integers below 12 that are multiples of either 2 or 4; the sum of 2, 4, 6, 8, 10 is 30.

Write a function **`mysum(a,b,limit)`** that accepts three arguments: a, b and limit. The arguments a and b are integers greater than zero and lesser than limit. The function **`mysum(a,b,limit)`** should return the sum of all the multiples of a or b, the multiples being lesser than limit. If the user enters a or b to be less than zero or non-integers, the function should return the error message "Wrong input" as a string.

Sample tests:	**Output:**
`print mysum(3, 5, 10)`	`23`
`print mysum(2,4,12)`	`30`
`print mysum(3,3,15)`	`30`
`print mysum(7,9,100)`	`1266`

```
print mysum(21,34,10000)            3783486

print mysum (0, 5, 10)              'Wrong input'

print mysum(0.5, 5, 10)             'Wrong input'

print mysum(3, 'x', 10)             'Wrong input'

print mysum(2, 3, 0)                0
```

Q.5 [10 points]

Write a function called **`get_students(students, course)`** which takes in a list and a string. The input list is made up of a sequence of binary tuples, each with a student name and a list of courses that the student has enrolled in. The second argument is a string containing the name of a course. Your function should return a list of the names of students who are enrolled in that course. If no students are taking the course the function should return an empty list. For example, see the test code and expected output below:

Test Code:

```
students = [("Alan", ["CompSci", "Physics", "Math"]),
            ("Justin", ["Math", "CompSci", "Stats"]),
            ("Edward", ["CompSci", "Philosophy", "Economics"]),
            ("Margaret", ["InfSys", "Accounting", "Economics",
           "CommLaw"]),
            ("Philip", ["Sociology", "Economics", "Law", "Stats",
           "Music"]),
            ("Mary", ["Math", "CompSci", "Stats"]),
            ("Vera", ["CompSci", "Philosophy", "Economics"]),
            ("Mike", ["InfSys", "Accounting", "Economics", "CommLaw"]),
            ("Donna", ["Sociology", "Economics", "Law", "Stats"])]
print get_students(students, 'Philosophy')
print get_students(students, 'History')
print get_students(students, 'Math')
print get_students(students, 'CompSci')
```

Expected Output:

```
['Edward', 'Vera']
[]
['Alan', 'Justin', 'Mary']
['Alan', 'Justin', 'Edward', 'Mary', 'Vera']
```

Q.6 [Total: 30 points]

SUTDBook is a social network website founded by one of SUTD graduates. They are currently hiring some software engineers to develop an algorithm to suggest new 'friends' to their user. Your task in this question is to build this new friends suggestion algorithm.

Network of users can be represented as a graph of connected nodes where each user is a node. The connection between two nodes states a friend relationship between the two users.

Overall Test Cases:

To test, we provide several text files: facebook_less.txt, studbook1.txt, and sutdbook2.txt.

Test Code:

```
f=open('facebook_less.txt','r')
nodes= get_nodes(f)
G= create_graph(nodes)
print 'Friends of 1 from facebook_less.txt'
print get_friends(G,1)
print 'Suggested new friends for 1'
print suggested_new_friends(G,1)
f.close()

f=open('sutdbook1.txt','r')
nodes= get_nodes(f)
G= create_graph(nodes)
print 'Friends of 0 from sutdbook1.txt'
print get_friends(G,0)
print 'Suggested new friends for 0'
print suggested_new_friends(G,0)
f.close()

f=open('sutdbook2.txt','r')
nodes= get_nodes(f)
G= create_graph(nodes)
print 'Friends of 0 from sutdbook2.txt'
print get_friends(G,0)
print 'Suggested new friends for 0'
print suggested_new_friends(G,0)
f.close()
```

Expected output:

```
Friends of 1 from facebook_less.txt
[0, 322, 133, 73, 299, 236, 48, 53, 54, 92]
Suggested new friends for 1
([25, 88], 4)

Friends of 0 from sutdbook1.txt
[1, 2, 3]
Suggested new friends for 0
([48, 53], 1)

Friends of 0 from sutdbook2.txt
[1, 2, 3]
Suggested new friends for 0
([5], 3)
```

(a) [5 points]

Write a function called **`get_nodes(fid)`** which takes in a file object as its input arguments and outputs a list of tuples. Each tuple shows a friend connection between two users and each user is represented by an **integer**. A sample of the text file can be seen below:

```
#extract of sutdbook1.txt
0     1
0     2
0     3
1     48
1     53
```

The first three lines of the above text file means that user 0 is a friend of user 1, 2, and 3. The last two lines means that user 1 is a friend of user 48 and 53.

Test code:

```
f=open('sutdbook1.txt','r')
result=get_nodes(f)
print result
```

Expected output:

```
[(0,1),(0,2),(0,3),(1,48),(1,53)]
```

(b) [10 points]

Note: If you find this part too difficult, you can do part (c) first. Each part can be done independently.

Write a function **`create_graph(nodes)`** which takes in a list of tuples and returns a graph of friend connection as a dictionary. The list of tuples is obtained from the output of **`get_nodes(fid)`** in part (a) and represents a friend connection between two users. The output of the function is a dictionary with each user as a key. The value is also a dictionary that contains key-value pair of each of the user's friends. For example, if the input is

```
[(0,1),(0,2),(0,3),(1,48),(1,53)]
```

The expected output is:

```
{0:{1:1, 2:1, 3:1},
 1: {0:1, 48: 1, 53:1},
 2: {0:1},
 3: {0:1},
 48: {1:1},
 53:{1:1}}
```

In the above output, we can see that user 0 has three friends, i.e. users 1, 2, and 3. Notice that each of his friends is a key in the inner dictionary with a value of 1. Similarly, user 1 has three friends, i.e. users 0, 48, and 53, while user 2 only has one friend, i.e. user 0, and so on.

Note that the values of the inner dictionary are always 1. Even though, we do not use this value, the data structure in this problem uses a dictionary because it is easier to operate rather than a list. For example, to remove friendship between user 0 and user 1, we can just type:

`del G[0][1]` and **`del G[1][0]`**.

(c) [5 points]

Write a function called **`get_friends(G,node)`** that takes in two arguments. The first argument is a dictionary that contains the network of friends and the second argument is the node (or user) of interest. The function **`get_friends(G,node)`** returns a list of friends that particular user has. For example, for a given dictionary:

```
G={0:{1:1, 2:1, 3:1},
 1: {0:1, 48: 1, 53:1},
 2: {0:1},
 3: {0:1},
 48: {1:1},
 53:{1:1}}
```

`get_friends(G,0)` returns users 1, 2, and 3, while **`get_friends(G,2)`** returns user 0.

Test code:

```
print get_friends(G,0)
print get_friends(G,1)
print get_friends(G,2)
```

Expected output:

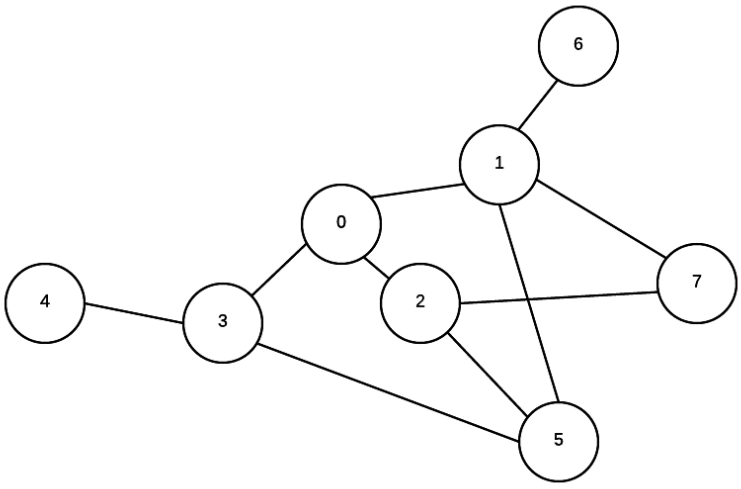
```
[1, 2, 3]
[0, 48, 53]
[0]
```

(d) [10 points]

Write a function called **`suggested_new_friends(G,node)`** that takes in a dictionary describing the friends network and an integer representing a user. The function returns a list of suggested new friends for the input user. The algorithm to suggest new friends works as follows:

1. Get all the friends of the current user.
2. Get the friends of the current user's friends that are not currently the current user's friends.
3. Rank these friends of friends by counting how many current user's friends have connection with them.
4. Return a list of friends with the highest count and the number of count.

For example, the image below shows a graph of a network of friends. You can get this graph from **sutdbook2.txt**.



Which can be described by the following dictionary

```
G={0:{1:1, 2:1, 3:1}, 1:{0:1, 5:1, 6:1, 7:1}, 2:{0:1, 5:1, 7:1},
3:{0:1, 4:1, 5:1}, 4:{3:1}, 5:{1:1, 2:1, 3:1}, 6:{1:1}, 7:{1:1, 2:1}}
```

If we run the function **`suggested_new_friends(G,0)`**, we are interested in finding new friends for user 0.

- Step 1 produces the friends of user 0:
  - [1, 2, 3]
- Step 2 produces the common friends of user 0's friends:
  - User 5 is a friend of users 1, 2, and 3
  - User 7 is a friend of users 1 and 2
  - User 4 is the only friend of user 3
  - User 6 is the only friend of user 1
- Step 3 produces the count for each possible new friends:
  - User 5 has 3 friends of user 0
  - User 7 has 2 friends of user 0
  - User 4 has 1 friend of user 0
  - User 6 has 1 friend of user 0
- Step 4 produces the suggested new friends for user 0.
  - User 5 has the highest count (i.e. 3 friends of user 0) so user 5 is the suggested new friend. The function should return a tuple, i.e. ([5], 3). Since there is only one user with three counts, the list contains only one element, i.e. user 5.

Test code is provided in beginning of this file under **Overall Test Cases**.

Q.7 [15 points]

Given n, a nonnegative integer (i.e., n>=0), write an efficient program to find the number of **nonnegative integer solutions** to:

x1 + x2 + x3 + x4 + x5 = n

Here, xi is an integer and xi >=0

Specifically, write a Python function: **`num_of_sol(n)`**

Input argument n is a nonnegative integer, i.e., n>=0

The function returns the number of the possible solution: x1,x2,x3,x4,x5

Note that you need to come up with an efficient way to perform the counting. You function needs to handle large n, e.g. n = 150, in less than 20 seconds. A brute-force solution that tests every possible solution would not satisfy the requirement.

Test case:

```
print num_of_sol(3)
print num_of_sol(4)
```

Output:

```
35
70
```

Note: when n = 3, there are 35 possible solution (x1,x2,x3,x4,x5), as follows:

0,0,0,0,3	0,0,0,1,2	0,0,0,2,1	0,0,0,3,0	0,0,1,0,2	0,0,1,1,1	0,0,1,2,0	0,0,2,0,1	0,0,2,1,0
0,0,3,0,0	0,1,0,0,2	0,1,0,1,1	0,1,0,2,0	0,1,1,0,1	0,1,1,1,0	0,1,2,0,0	0,2,0,0,1	0,2,0,1,0
0,2,1,0,0	0,3,0,0,0	1,0,0,0,2	1,0,0,1,1	1,0,0,2,0	1,0,1,0,1	1,0,1,1,0	1,0,2,0,0	1,1,0,0,1
1,1,0,1,0	1,1,1,0,0	1,2,0,0,0	2,0,0,0,1	2,0,0,1,0	2,0,1,0,0	2,1,0,0,0	3,0,0,0,0	

Submit your code to Tutor. **You need not print out all possible combination of output. You only need to return the number of solutions.** Note that, for this question, the test cases in Tutor have different weights. Ignore the points given out by Tutor.

(Hint: First, you may consider a simpler problem: Find the number of solutions to x1+x2=n. Then, try to generalize your reasoning to the current problem: x1+x2+x3+x4+x5=n)

End of Exam Paper

[this page is left blank]

[this page is left blank]