Lab Test

5 Oct 2018 (Fri, Week 7) 3:30-7:30pm

General Instructions:

- There are 4 questions in this test to be completed individually; all questions are mandatory. This test is scored over 20 marks and is worth 10% of your final grade.
- You are expected to complete it within 2 hours, but you are allowed up to 4 hours to complete this exam. You are encouraged to leave early once you have checked your work.
- This is an open-book, open-Internet test: you may refer to any hardcopy or softcopy material already on your laptop. You may search the Internet.
- But **you may NOT communicate directly or indirectly with anyone** (including non-candidates) throughout the duration of the whole lab test. Examples of communication include: posting anything (request for hints, partial solutions, questions etc.) on Internet forums, corresponding via email, instant messaging or social media, sharing a folder or writing to a shared folder. **This is your last warning.**
- For this test, your team ID is your name (i.e. you are the only member in your team).
- Shut down all email/Instant Messaging (IM) clients on your laptop (Outlook, Telegram, What's App browser etc.). You are not allowed to use any form of IM or your mobile phone during the test.
- **Before you leave the room:**
  (i) Upload a zipped up file containing your solutions (the four .py files) to eLearn. This serves as a backup and will not be marked under normal circumstances.
  (ii) Tell your invigilator your email ID, and
  (iii) Leave the room quietly.
  Your account on red.smu.edu.sg will be suspended once you leave the exam room.

Submission Instructions:

- You will work on the problems on your personal laptop, and submit your solutions at http://red.smu.edu.sg. Only your submissions to the server will be marked, so you should check that you have submitted all your solutions correctly before leaving the room. No written submission is required.
- You can submit your solutions to red as many times as you wish, but the final submission before the test ends will be taken as your final submission.
- You are only supposed to submit these 4 files: **lt1.py**, **lt2.py**, **lt3.py** and **lt4.py**. Do not change the function signature in these files; you are only allowed to replace the statements in the functions in these files.
- You are advised to submit your py file for each question to red before moving on to the next question.

Q1 [4 marks] – Loops (no quality score, no time)

Write a function that takes in two integers (i and j), and returns the product of all the integers between i and j (**including** i, and **excluding** j):

```
    def get_product(i, j):
```

You can assume that **j** will always be bigger than **i** by at least 2. You are given some test cases in **lt1_main.py** to validate your function.

Test cases:

#	i	j	get_product(i, j)
1	-5	-1	-5 x -4 x -3 x -2 = 120
2	5	10	5 x 6 x 7 x 8 x 9 = 15120
3	-2	4	-2 x -1 x 0 x 1 x 2 x 3 = 0
4	1	3	1 x 2 = 2

Scoring:

- Your function must complete running with 30 seconds on the server
- Your function must return a correct solution.
- Quality score: N/A. Will appear as "1.0" if solution is correct.
- Time score: N/A

Q2 [4 marks] – Iteration (no quality score, no time)

Write a function that takes in an age, and a 2D list of employees ([[ID, birthyear], [ID, birthyear]…]), and return the number of employees in the list who are **older** than the specified age (you can assume that the current year is 2018).

```
    def get_count(age, ewby_list):
```

(ewby stands for "employee with birth year")

For testing, you are given 2 lists of employees with birthyear in **lt2_main.py**. You can simply call **get_ewby_list1()** and **get_ewby_list2()** to retrieve them. List1 has 100 employees, while list2 has 10,000 employees.

Test cases:

#	age	ewby_list	get_count(age, ewby_list)
1	50	list1	30
2	25	list1	74
3	45	list2	4239
4	32	list2	6563

Scoring:

- Your function must complete running with 30 seconds on the server
- Your function must return a correct solution.
- Quality score: N/A. Will appear as "1.0" if solution is correct.
- Time score: N/A

Q3 [5 marks] – Recursion (no quality score, no time)

I have **n** identical coins which I want to give you in groups. Each time, I can only give you 1, 2 or 3 coins in a group. How many different ways can I give you all my coins?

Example: if **n** is 3 (meaning I have only 3 coins), there are 4 ways I can give them to you:

- 1 + 1 + 1	(3 groups)
- 1 + 2	(2 groups)
- 2 + 1	(2 groups)
- 3	(1 group)

Example: if **n** is 4, there are 7 ways I can give them to you:
- 1 + 3
- 1 + 1 + 1 + 1
- 1 + 2 + 1
- 1 + 1 + 2
- 2 + 1 + 1
- 2 + 2
- 3 + 1

Write a function that uses recursion to compute the number of ways I can give you the coins with **n** as an input parameter. You can assume that **n** will always be bigger than 0.

```
def no_of_ways(n):    # returns an integer
```

Test cases:

#	n	no_of_ways(n)
1	5	13
2	10	274
3	11	504
4	15	5768

Scoring:
- Your function must complete running with 30 seconds on the server
- Your function must return a correct solution. <u>**Your solution must be recursive**</u>; your submission will be manually checked that it uses recursion to obtain the answer.
- Quality score: N/A. Will appear as "1.0" if your solution returns the correct answers.
- Time score: N/A

Q4 [7 marks] - Devise an algorithm to get a good quality score

This question is an extension of lab 3. Instead of selecting only five Twitter users who will broadcast the advertisement, management has approved a budget to pay many more Twitter users to send the advertisement. However this time round, management wants EVERY user to get the message, but at the same time, pay as few Twitter users as possible. So, the new goal for you is to select the smallest number of Twitter users who need to send the advertisement so that everyone will get it.

Requirements:

- Write a function called **select_set**
  ```
  def select_set(followers):
  ```
  This function takes in 1 parameter **followers** (a 2D list of Twitter users) and returns a list of user IDs (integers) selected by your algorithm who will broadcast the advertisement. **len(followers)** will give you the total number of Twitter users in the Twitter universe.
- The objective is to make the list of IDs returned by **select_set()** as small as possible such that every user gets the advertisement.

Your solution will be checked using this function: **all_users_got_message**. This function takes in your list of selected Twitters (**selected**) and the original **followers** 2D list, and returns **True** (if every Twitter user would have gotten the message) or **False** (if at least one Twitter user has not gotten the message). If **all_users_got_message** returns **True**, your solution's quality will depend on the number of users in your selected set (i.e. **len(selected)**).

Example

Refer to figure 1 that shows a simple scenario with only 13 users in the Twitter universe (arrow means "is following" – e.g. user 7 is following user 6). The corresponding CSV file is **case10.csv**.

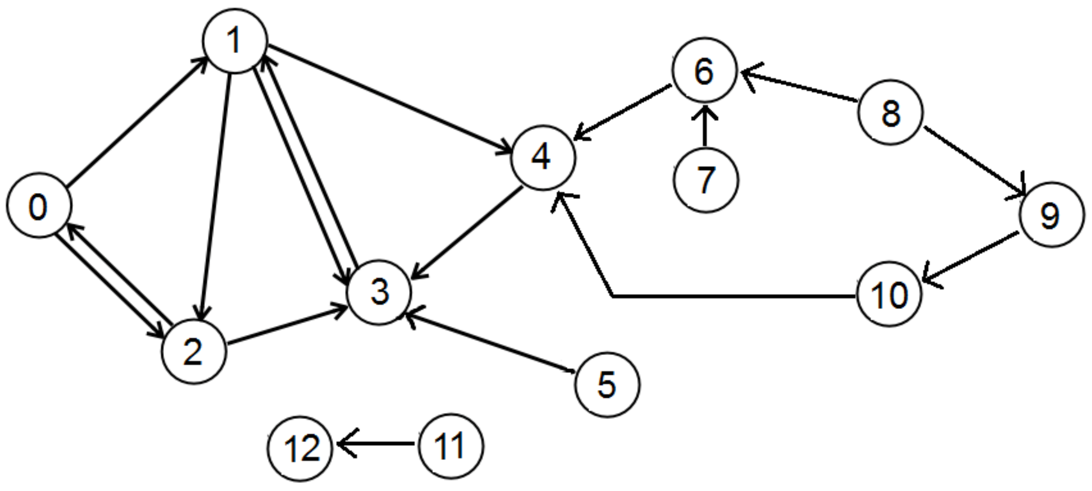

*Figure 1: graph showing users in **case10.csv***

Note that user 11 has no follower, and user 12 has only one follower (user 11).

The following table is derived from figure 1:

User ID	ID of Direct Followers
0	2
1	0, 3
2	0, 1
3	1, 2, 4, 5
4	1, 6, 10
5	
6	7, 8
7	
8	
9	8
10	9
11	
12	11

Since all 13 users MUST get the message, a naive solution will be to select all 13 users who will broadcast the advertisement. But that's an expensive solution because the company will have to pay all 13 users. A better solution will be to select (and pay) only these users: **[0, 4, 9, 7, 3, 12]**. According to the next table, by selecting these six users to tweet the advertisement, everyone (users 0-12) will get the advertisement.

User ID	Got advertisement?
0	Yes – selected sender
1	Yes – get from user 4 or 3
2	Yes – get from user 0 or 3
3	Yes – selected sender
4	Yes – selected sender
5	Yes – get from user 3
6	Yes – get from user 4
7	Yes – selected sender
8	Yes – get from user 9
9	Yes – selected sender
10	Yes – get from user 4
11	Yes – get from user 12
12	Yes – selected sender

Assuming that your **select_set()** function returns **[0, 4, 9, 7, 3, 12]**, here's what will happen:

```
>>> followers = [[2], [0,3], [0,1], [1,2,4,5], [1,6,10], [], [7,8],
[], [], [8], [9], [], [11]]

>>> s = select_set(followers)
[0, 4, 9, 7, 3, 12]

>>> all_users_got_message(s, followers)
True
```

```
>>> score = len(s)     # the smaller the better
6
```

However, if your **select_set()** returns **[1, 2, 3]**, here's what will happen:

```
>>> t = select_set(followers)
[1, 2, 3]

>>> all_users_got_message(t, followers)
False
```

Work it out manually: if only users **[1, 2, 3]** send the advertisement, several users (6, 7, 8, 9, 10, 11 and 12) will not get the message. **[1, 2, 3]** is hence considered a wrong answer, and will not be scored.

You are given the following files:

File name	Description	Comments
lt4.py	Contains the **select_set** function that you will write.	You need to modify and submit this file. This is the only file that you may submit. Do not modify the file name or the function signature in the file.
lt4_main.py	Calls the **select_set** function you will write in **lt4.py**.	Do not submit this file; use it to check if your **select_set** function in **lt4.py** works before submitting it to the Submission Server.
lt4_utility.py	Contains function(s) used by **lt4_main.py**.	Do not submit this file.
data folder containing CSV files	Contains text files read by **lt4_main.py**. The 2D list of user IDs used in the test cases are read from CSV files in this folder.	Do not submit these files. **case10.csv** contains the Twitter network described in figure 1. The other CSV files contain more complex networks of more users. Do take a look at **case11-12.csv** using either Microsoft Excel or a text editor. The test case on the Submission Server will use data of similar characteristics. There are several users who have no followers, and several users who do not follow anybody.

The server uses a different test case, which is not given to you.

Scoring:

- Your function must complete running with 5 minutes on the server
- Your function must return a "correct" solution – i.e. all users must get the advertisement if your selected users send out the advertisement.
- Quality score: will be the number of selected users returned by your function. The smaller the score, the better it is.
- Time score: you may be given some credit if the time taken by your function is faster. However, quality score is much more important than time taken. So do focus on quality.
- **The "quality score" on the scoreboard at red is merely a guide; after the exam ends, your code may be executed again using different data sets to derive the final score.**

~End