Assignment #1

In this assignment, you are asked to demonstrate a program which uses the container class in Python, i.e. *list* and *dict*, to automatically find all the duplicated book items in the UTAR library collection. You can only use the Python built-in functions to perform the task and other kinds of advanced modules, such as *pandas* and *numpy*, will immediate lead to zero mark.

The outcome will be one single CSV file named as ***duplication.csv***. The format of the CSV file is described in ***duplication_example.csv***. Each row in the CSV file contains all the information of the same **TITLEs** from different databases **DBs**; i.e. the information are the TITLE, DB, ISSN, e-ISSN, ISBN, and e-ISBN numbers.

The input file is ***library-titles.xlsx***. Each row of the input file contains the related information of one single book item. An ISSN or e-ISSN being considered must have 8 digits; some are separated by '-' while some are not. An ISBN or e-ISBN being considered must be either 10 digits or 13 digits; some are separated by '-' while some are not. Hence, you need to reset those which are not conforming to the aforementioned digit formats to null, instead of dropping, before comparison.

Conditions for a duplication:

1) Exact match of any of the **ISSN**, **e-ISSN**, **ISBN**, and **e-ISBN** and exact match of the **TITLE** name from two or more book items.
2) Such matches must come from different **DB**.; i.e. multiple (same) book items from the same **DB** must be merged as one book item before matching.
3) Only output one record for the duplicated titles in ***duplication.csv***; i.e. permutation should not be excluded. See below:

Scenario 1	TITLE	DB	ISSN	...	TITLE	DB	ISSN	...	TITLE	DB	ISSN	...
	title1	A	1234-4567	...	title1	B	1234-4567	...	title1	A	1234-4567	...

Scenario 2	TITLE	DB	ISSN	...	TITLE	DB	ISSN	...
	title1	B	1234-4567	...	title1	A	1234-4567	...

Scenario 3	TITLE	DB	ISSN	...	TITLE	DB	ISSN	...
	title1	A	1234-4567	...	title1	B	1234-4567	...

All of the scenarios above are considered the same description of duplication and your output should use *scenario 3* where *unqiue* duplicates are *sorted* alphabetically by **DB**.

Marking scheme: (Total 20 marks)

1. Use a flowchart to explain how you plan to solve the problem. (3 marks)
2. Give at least three different scenarios of duplications exist in the data file. (3 marks)
3. Demonstrate modular programming technique to implement the flowchart that was designed to answer marking scheme 1). (2 marks)
4. Show effective use of *list comprehension* or *dictionary comprehension*. (4 marks)

5. The solution cannot have any 3- or more levels of nested loops within the main or any functions. (2 marks)
6. The solution can run to completion without error messages. (2 marks, not a dummy run)
7. Output filename and its contents must conform with the given formats. (2 marks)
8. With clear and sufficient comments in the program to explain your design. (2 mark)

You are not allowed to change the format of any inputs for your convenience. Also, you should not assume any other inputs from the keyboard or files unless otherwise specified later.

Submission

- The deadline is **6pm on 17th March 2020 (Tuesday)**
- **NO LATE submission** is accepted; i.e. zero mark.
- One submission for one group is sufficient.
- Do NOT submit input files.
- Only submit your solution in Python along with your output file by emailing them to laiac@utar.edu.my with email subject equal to
  - **UEEN1043 2020 A1 G[***group number***]**
  - e.g. UEEN1043 2020 A1 G[13] for group #13.
- Your Python file should follow the naming format below:
  - **G[***group number***]_A1.py**
  - e.g. G[13]_A1.py for group #13.
- Failure to conform to the naming format above will result in **10% deduction of your total mark**