LAB 8 : Graph

[CO3]

Instructions for students:

- Complete the following methods.
- You may use Java / Python to complete the tasks.
- DO NOT CREATE a separate folder for each task.

NOTE:

- **YOU CANNOT USE ANY OTHER DATA STRUCTURE OTHER THAN ARRAY UNLESS MENTIONED IN THE QUESTION.**
- **YOUR CODE SHOULD WORK FOR ANY VALID INPUTS.**

Python List, Negative indexing and append() is STRICTLY prohibited unless mentioned in the question

Lab Tasks: 4 tasks (0~1b)

Assignment Tasks: 6 tasks (2a~4b)

Total Assignment Mark: 6*5=30

Dear Students, you have been given instructions and driver code for the majority of the labs. For this lab no driver code will be given. You will develop everything (necessary functions, class, driver code, etc.) in your preferred language (Java or Python).

To solve the problems, draw a graph (https://graphonline.top) according to your preference. However, your graph must have a minimum of **7 vertices** and a minimum of **12 edges**.

Note: Solve all the problems for both adjacency matrix representation and adjacency list representation. You can write different functions for each task to make your life easier.

Task 0: [Lab Task]

Represent the Graph you just drew in your code. Using both **Adjacency Matrix (Task0a)** and **Adjacency List (Task0b)**.

Task 1: [Lab Task]

Given an undirected, unweighted graph as input, write a function to find the vertex with maximum degree and return the degree of that vertex. Solve it for Adjacency Matrix **(Task1a)** & Adjacency List **(Task1b)**.

Task 2: [Assignment Task]

Given an undirected, edge-weighted graph as input, write a function to find the vertex whose sum of edge weights is maximum. Solve it for Adjacency Matrix **(Task2a)** & Adjacency List **(Task2b)**.

Task 3: [Assignment Task]

Solve Task 1 and Task 2 for directed, edge-weighted graphs considering only outgoing edges. Solve it for Adjacency Matrix **(Task3a)** & Adjacency List **(Task3b)**.

Task 4: [Assignment Task]

Write a function that will receive a directed weighted graph and convert it to an undirected weighted graph. Consider the weight of the edges will be unchanged. If there's a case where you have a bidirectional edge then you can sum up their weights. Solve it for Adjacency Matrix **(Task4a)** & Adjacency List **(Task4b)**.