Department of Electrical and Software Engineering
Schulich School of Engineering

ENSF 694 - Principles of Software Development II
Summer 2023

Lab 9 – July 31, 2023
Topic: Graphs

- All codes must be complete and compile without any errors.
- The codes should work for not only the given sample inputs but also any inputs of the same data types.
- **Submission: github link of the codes in the d2l dropbox 'Lab#09_Jul31' and push your code in the github classroom repository**
    - Go to this link - https://classroom.github.com/a/SsDTPtib
    - Refresh and accept the Lab9 link, Clone the repository and then push your code, Then submit the github link to the d2l dropbox 'Lab#09_Jul31'

Lab Tasks

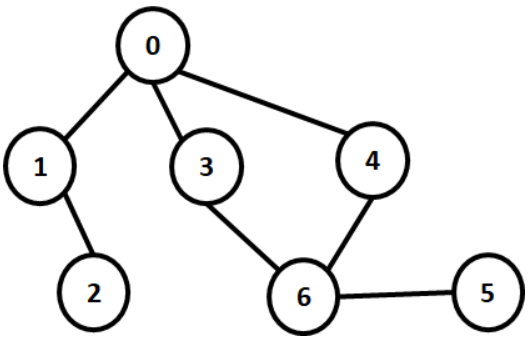
15 marks =
05 marks for input-output format +
05 marks for implementing graph adjacency matrix and queue +
05 marks for implementing BFS.

Q1.

1. Take the number of nodes, number of edges and endpoints of each edge of an undirected unweighted simple graph as inputs from the user.
2. Then create an adjacency matrix for the graph and show the matrix.
3. Apply Breadth First Search (BFS) for the graph traversal.
4. Use your own queue class and methods for implementing the queue needed for BFS (you can use the queue class you already implemented for labs and/or assignments).
5. Add some extra print statements to the queue operations and BFS operations to show the execution of enqueue, dequeue and nodes visited for adjacent nodes as the sample output to show how the BFS algorithm is working on the set of nodes and edges.
6. Finally show the visit sequence of the nodes of the input graph.
7. You are not allowed to use existing java.util.LinkedList, java.util.queue, or any of their built in methods for creating and accessing your queue.

Sample Run of the Code:

The input graph for this sample execution looked like this-



```
Enter number of nodes:
7
Enter number of edges:
7
Enter the edges in u v order where u and v are the endpoints:
0 1
0 3
0 4
1 2
3 6
4 6
6 5

Adjacency matrix for the undirected grpah:
0 1 0 1 1 0 0
1 0 1 0 0 0 0
0 1 0 0 0 0 0
1 0 0 0 0 0 1
1 0 0 0 0 0 1
0 0 0 0 0 0 1
0 0 0 1 1 1 0

BFS visit of the grpah:
Enqueued node: 0
Dequeued node: 0
Current node being processed for adjacent nodes: 0
Unvisited adjacent nodes exists for 0
Enqueued node: 1
Unvisited adjacent nodes exists for 0
Enqueued node: 3
Unvisited adjacent nodes exists for 0
Enqueued node: 4
No more unvisited adjacent nodes exists for 0
Dequeued node: 1
Current node being processed for adjacent nodes: 1
Unvisited adjacent nodes exists for 1
Enqueued node: 2
No more unvisited adjacent nodes exists for 1
Dequeued node: 3
Current node being processed for adjacent nodes: 3
Unvisited adjacent nodes exists for 3
Enqueued node: 6
No more unvisited adjacent nodes exists for 3
Dequeued node: 4
Current node being processed for adjacent nodes: 4
No more unvisited adjacent nodes exists for 4
Dequeued node: 2
Current node being processed for adjacent nodes: 2
No more unvisited adjacent nodes exists for 2
Dequeued node: 6
Current node being processed for adjacent nodes: 6
Unvisited adjacent nodes exists for 6
Enqueued node: 5
No more unvisited adjacent nodes exists for 6
Dequeued node: 5
Current node being processed for adjacent nodes: 5
No more unvisited adjacent nodes exists for 5

BFS visit sequence of the grpah:
0 1 3 4 2 6 5
```