CS222: Assignment 9 - Graph algorithms Part 1

1. Submission deadline: Sunday, 25 April at 11:59 pm.
2. Follow good coding practices to gain more marks.
3. The assignment can be submitted in groups of size ≤ 3.
4. Submit two `.cpp` files and two `.pdf` files with the output.
5. Write the names and roll numbers of the students at the top of each file.
6. The files should be called

   `directedgraph_firstRollNumber_secondRollNumber_thirdRollNumber.cpp`,

   `directedgraph_firstRollNumber_secondRollNumber_thirdRollNumber.pdf`.

   `undirectedgraph_firstRollNumber_secondRollNumber_thirdRollNumber.cpp`,

   `undirectedgraph_firstRollNumber_secondRollNumber_thirdRollNumber.pdf`.
7. Anusha Devulapally is the TA for this assignment.
8. In this assignment, you will implement the graph algorithms that you have learnt in the class.

---

1. *Directed graph, adjacency list:* You will work with only directed graphs in this question. Create a class `DirectedGraph`.
   (a) (10 points) Take a directed graph from the user as an input. The interaction with the user should look like this example.

   ```
   Enter the number of vertices in your graph: 16
   Do you want to enter more edges? (y/n): y
   Enter the edge: 2 5
   Edge (2, 5) added.
   Do you want to enter more edges? (y/n): y
   Enter the edge: 3 7
   Edge (3, 7) added.
   Do you want to enter more edges? (y/n): y
   Enter the edge: 8 15
   Edge (8, 15) added.
   Do you want to enter more edges? (y/n): n
   ```

   Store the graph as an adjacency list. Once you have the graph G, you are ready to call the following public functions of your class `DirectedGraph`

   (b) (10 points) `G.DFS (int n)` Output the set of reachable vertices from vertex n in the sequence that they are visited. Implement it using recursion.

(c) (10 points) `G.BFS (int n)` Output the distance of each vertex from vertex n. If a vertex is not reachable, output `-1`.

(d) (15 points) `G.linearization()` Outputs a linearization of G if the graph is a DAG. Otherwise prints the message `"The graph G is not a DAG."`. Do not destroy the original graph. Use a local copy.

2. *Undirected graph, adjacency matrix:* You will work with only undirected graphs in this question. Create a class `UndirectedGraph`.

(a) (5 points) Take an undirected graph from the user as an input. The interaction with the user should look like this example.

```
Enter the number of vertices in your graph: 16
Do you want to enter more edges? (y/n): y
Enter the edge: 2 5
Edge (2, 5) added.
Do you want to enter more edges? (y/n): y
Enter the edge: 3 7
Edge (3, 7) added.
Do you want to enter more edges? (y/n): y
Enter the edge: 8 15
Edge (8, 15) added.
Do you want to enter more edges? (y/n): n
```

Store the graph as an adjacency matrix. Once you have the graph G, you are ready to call the following public functions of your class `UndirectedGraph`

(b) (10 points) `G.DFS (int n)` Output the set of reachable vertices from vertex n in the sequence that they are visited. Implement it without using recursion.

(c) (10 points) `G.BFS (int n)` Output the distance of each vertex from vertex n. If a vertex is not reachable, output `-1`.