COMP108 Data Structures and Algorithms
Lab Exercises (Week 8)

Due: 25 March 2022, 5:00pm

Information

- Submission: Submit the file COMP108W08.java to SAM
  https://sam.csc.liv.ac.uk/COMP/CW_Submissions.pl?qryAssignment=COMP108-18
  Late submission is only accepted until Monday 9:00am.
- Submission of lab/tutorial exercises contributes to 10% of the overall module mark. Submission is marked on a pass/fail basis - you will get full marks for submitting a *reasonable attempt*.
- Individual feedback will not be given, but solutions will be posted promptly after the deadline has passed.
- These exercises aim to give you practices on the materials taught during lectures and provide guidance towards assignments.
- Relevant lectures: **Lecture 19**
- You can refer to the guidance on how to use the web-based IDE https://ide.cs50.io/.

1. **Programming — Preparation**
   (a) Download three java files "COMP108W08App.java", "COMP108W08.java" from Canvas via the link "Labs & Tutorials" → "Week 8".
   (b) Compile the programs by typing first **javac COMP108W08.java** and then **javac COMP108W08App.java**. There should be two files created: COMP108W08.class and COMP108W08App.class.
   (c) Run the program by typing **java COMP108W08App**
   (d) **Every time you have edited COMP108W08.java, you have to (i) recompile by javac COMP108W08.java and then (ii) run by java COMP108W08App.**

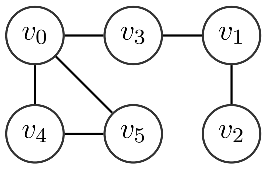
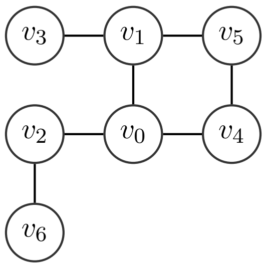
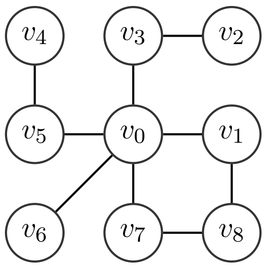
2. **Graph** This week we will work with graph.
   - An adjacency matrix represents a graph by representing the edges between vertices.
   - An entry of the adjacency matrix $M[i][j]$ is 1 if vertex i and vertex j have an edge between them and 0 otherwise.

3. **Task 1: Degree of a graph.** In COMP108W08.java, the method findDegree() is expected to find the **degree** of the graph, which is the **maximum** degree of the vertices and the degree of a vertex is the **number of neighbours** it has. The method takes in as parameter a two dimensional array `adjMatrix[][]` for the adjacency matrix and an integer `gSize` for the size of the graph (number of vertices).

   Complete the method (without changing its signature) and test it using test cases stated at the end of the document.

4. **Task 2: Distance between two vertices.** The method distance() is expected to determine for two vertices if `v1` and `v2` are at distance 1, distance 2, or not connected at distance 2.

Test cases:

Test case	Graph	Input	Output
#1 graph1.txt		6 0 0 0 1 1 1 0 0 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 1 0 4 1 0 1 1 5 1 1 1 1 -1	Degree: 3 Distance between vertex 0 and 4: 1 Distance between vertex 0 and 1: 2 Vertex 5 and 1 are not distance 2 apart. Vertex 1 and 1 are not distance 2 apart.
#2 graph2.txt		7 0 1 1 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 2 6 1 0 6 1 5 2 -1	Degree 3 Distance between vertex 2 and 6: 1 Distance between vertex 0 and 6: 2 Vertex 5 and 2 are not distance 2 apart.
#3 graph3.txt	 v_4 v_3 v_2 v_5 v_0 v_1 v_6 v_7 v_8	9 0 1 0 1 0 1 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 4 5 1 1 3 1 5 8 -1	Degree: 5 Distance between vertex 4 and 5: 1 Distance between vertex 1 and 3: 2 Vertex 5 and 8 are not distance 2 apart.