1. [10 marks] Consider the following directed graph. Assume that you are using the adjacency list representation, and in the adjacency list for any vertex, the vertices that appear are stored in \textit{increasing} numerical order.

(a) You perform a BFS starting at vertex 7. List the reachable vertices, in the order in which BFS encounters them.

(b) Starting from scratch, you perform a BFS starting at vertex 3. List the reachable vertices, in the order in which BFS encounters them.
2. [10 marks] The goal of this exercise is to get you to think about how graphs should actually be implemented, at a low level.

Suppose you want to implement an undirected graph in a program, and you want to allow deletion of a given vertex \( v \). Deleting a vertex \( v \) means deleting \( v \) and also all edges incident on \( v \).

Describe, in English, how to modify the adjacency list representation of a graph presented in class, so that you can carry out this deletion operation in \( O(\text{deg} v) \) time, where \( \text{deg} v \) is the degree of \( v \) (the number of vertices adjacent to \( v \)). No need to provide complete pseudocode or justify correctness.

Note: do not worry about renaming the vertices, in order to ensure they are numbered 1, 2, 3, \ldots, \( n - 1 \) after deletion. You can keep the old numbering of vertices. Make any reasonable assumption you need and justify it.

Hint: you might have to use a linked list and some extra pointers in your graph representation.

3. [10 marks] Let \( G \) be an undirected graph. Show how to modify the BFS algorithm so that, if you run it starting from a source vertex \( s \), it not only computes the shortest-path distance from \( s \) to every reachable vertex \( v \), but also the total number of shortest paths from \( s \) to \( v \).

Hint: only very minor modifications are needed.

Explain in English, give pseudocode, justify correctness, and state the running time.

4. [Bonus question: 5 marks extra credit only] Read the discussion of the CHECK-SEMIGROUP algorithm, using BFS, in the course notes at https://www.student.cs.uwaterloo.ca/~cs341/lecture14.html. Implement the algorithm in any language, and, using your program, find two 5 \( \times \) 5 integer matrices, with entries in \( \{0, 1, -1\} \) that generate a finite semigroup of size at least 800. To get credit you only need to provide the matrices and the size of the semigroup generated, not the program.

An extra 5 marks will be given to the person who finds two matrices of the type described, generating the semigroup of largest finite order among all solutions presented.