This tutorial will cover the following topics:
- Directed acyclic graphs.
- Traversing and searching through graphs (explicitly and implicitly).
- Creating a list of the next states for a simple game (tic-tac-toe).

For all problems in this tutorial, you may only define helper functions using local or lambda.
Question 1: Directed Acyclic Graphs
Given a directed graph \( g \), write a function \( \text{dag?} \ g \) which produces true if \( g \) is a directed acyclic graph and false otherwise. You may assume that every node in \( g \) appears as an in-neighbour in \( g \), even if there are no outgoing edges from that node.

The graph \( g \), defined in Racket on the next slide. (This is the same as the example graph \( g \) from Module 18.)

The graph \( g2 \), defined on the next slide.
;; Examples:
(define g
  '((A (C D E))
    (B (E J))
    (C ())
    (D (F J))
    (E (K))
    (F (K H))
    (H ())
    (J (H))
    (K ()))))
(define g2
  '((A (C D))
    (B (E))
    (C ())
    (D (B))
    (E (D)))))
(check-expect (dag? g) true)
(check-expect (dag? g2) false)
Question 2: Next Moves in Tic-Tac-Toe
In the game of Tic-tac-toe, two players take turns placing either X’s or O’s onto a three-by-three grid. The player who places X’s will always go first. The first player to get three in a row (either vertically, horizontally, or diagonally) wins the game. Otherwise, the game will end in a draw when the board is full.

In this problem, we will model a Tic-tac-toe board using the following data definition:

```
;; A TicTacToeGame is a (listof (listof (anyof 'X 'O '_)))
;; where there are exactly 3 rows and 3 columns.
```

Write a function (next-games game) to produce a list of the next possible game states for a given Tic-tac-toe board.

```
;; Examples:
(define g1 '(((X O _) (O X _) (X X O)))
(check-expect (next-games g1) '(((X O O)
                               (O X _))
                             (X X O)))
((X O _)
  (O X O))
((X O _)
  (O X O)
  (X X O)))
```
**Extension Question: Finished Games**

Write a function `(finished? game)` to determine whether a given Tic-tac-toe game has ended. The game can end in two possible ways:

1. One player has achieved three in a row (either horizontally, vertically, or diagonally).
2. The board is full (none of the spaces are blank).

;; Examples:
(define f1 '((X O _) (O X _) (X X O)))
(define f2 '((X _ O) (_ X O) (X X O)))
(check-expect (finished? f1) false)
(check-expect (finished? f2) true) ; there are three O’s in col #3

**Hint:** Consider defining at least two the helper functions:
- `(filled? game)` consumes a game board and produces true if it is full of pieces and false otherwise.
- `(three-in-a-row? row piece)` determines whether there is a three-in-a-row of this piece in the current row

For a challenge, can you make your solution work for an NxN board of any size?