Overview

- Let’s take a look at an example first.
  - When you swipe Facebook or LinkedIn, there are always some suggestions regarding who you may know. Some may be your classmates, some may be from the same institute as you, and there are always some you have no idea who they are.
  - How could these social media find possible connections for you? Does anyone have thought about this?
  - Suppose we have a small social network. The nodes at the ends of each purple edge mean they are friends on Facebook. Now, we want to find possible connections. Here we define the connection to be friend’s friend.

- In this graph, Alice and Bob are friends, Bob and Carol are friends, so Alice and Carol have a connection. So as for Bob and Mary, Carol and James, Mary and Alice, James and Bob.
- Question: If you are going to implement an algorithm to find all possible connections in a social network, how would you do that?
- Show connection.pl, go through the code, run it, and show the output.

```prolog
/**
 * Social media connections
 *
 * Launch terminal
 * $ swipl
 * consult('connection.pl').
 * friend(alice, X).
 * connection(X, Y).
 * ;
 */
friend(alice, bob).
friend(bob, carol).
friend(carol, mary).
friend(mary, james).
friend(james, alice).

connection(X, Y) :- friend(X, Z), friend(Z, Y).
```
- Today, we are going to talk about logic programming.
- **Logic programming emerged as a distinct paradigm in the 1970's.**
- What makes logic programming distinct and cool is that when you programming with logic program languages, you only need to say what you want, not how you want it done.
- A program written in a logic programming language consists of:
  - a set of **facts** about objects and their relationships (friends in the connection.pl)
  - a set of **rules** about objects and their relationships (connections in the connection.pl)
  - a set of **queries** about objects and their relationship (connection(X, Y). on the terminal)