Requirements

- *Due date*: October 6, 2023 at 11:55PM. This is an individual assignment. This assignment is weighted 15% of the course grade.
- Each student submits three files: `assignment1.pdf`, `spiders.lp`, and `spidershortlegs.lp`. The file `assignment1.pdf` should contain the answers to questions 1.1, 1.2, 1.3, 2.1, 2.2, and 3.1, including any snippets of code necessary.

Spanning Spiders

Let $G = (V, E)$ be a graph. A *spanning tree* T is a subgraph of G with the following properties:

- T is a tree (i.e. it contains no cycles and is connected).
- T contains all of the vertices of G.

A *spider* S is a tree with at most one vertex in S whose degree is 3 or more. Such a vertex is called the *centre* of the spider. If S consists of no vertices with degree 3 or more, then any vertex can act as the centre of the spider. Finally, a *leg* of a spider is a path from the centre to a vertex of degree 1.

To get a better understanding of the terminology, consider the following example graph.

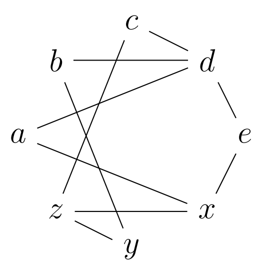

Figure 1: *An example graph.*

A possible spider is highlighted. In fact, the highlighted spider is a *spanning spider* because it is a spider *and* a spanning tree.

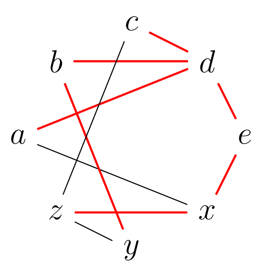

Figure 2: *A spider that is also a spanning tree.*

Given an input graph, our goal is to write an Answer Set Program to return a list of distinct spanning spiders (i.e. each stable model of our program corresponds to a distinct spanning spider). Not every graph has a spanning spider; if an input graph does not admit any spanning spiders, the corresponding program should return *unsatisfiable*.

Input format. An input file contains predicate instances of `vertex/1`, indicating the vertices of the graph and predicate instances of `edge/2`, indicating the edges of the graph. The example graph above is modelled with the following predicates.

```
vertex(a).
vertex(b).
vertex(c).
vertex(d).
vertex(e).
vertex(x).
vertex(y).
vertex(z).
edge(a, d).
edge(a, x).
edge(b, d).
edge(b, y).
edge(c, d).
edge(c, z).
edge(d, e).
edge(e, x).
edge(x, z).
edge(y, z).
```

Output format. Your program should compute a model containing a predicate `leg/2`, indicating which edges are included in the spanning spider. Each model should also contain a predicate `centre/1`, indicating which vertex is the centre of the spider.

Part I: Understanding the Problem

[1.1] Consider the spanning spider from Figure 2. Indicate the centre vertex and the list of edges that is included in the spanning spider.

[1.2] Provide ASP rules to define the `centre/1` predicate and ensure that in any model, *exactly* one vertex is selected as the centre.

[1.3] Provide a generator for the `leg/2` predicate.

Part II: Refining the Models

Up until now, you should have a set of models that consists of *exactly* one vertex as the centre of the spider and a set of leg edges. Our goal is to now remove some of these models that can form a spanning spider.

[2.1] One property required of spanning spiders is that every vertex should be reachable from the centre through leg edges. Introduce a derived predicate `reachable/1` that ranges over vertex names and is true when the corresponding vertex is reachable from the centre through leg edges. Use this predicate to define a constraint that ensures every vertex of the original graph is reachable from the centre through leg edges.

[2.2] The reachability of every vertex from the centre is a required property of spanning spiders, but it is not sufficient and we need to ensure other constraints are satisfied. Describe all the other constraints that need to be satisfied and write ASP rules to enforce these constraints.

We are now ready to write an ASP program that returns the set of spanning spiders, given an input graph.

Part III: Putting everything together

[3.1] Based on your answers to the above questions, write an ASP program `spiders.lp` that takes an input graph and outputs all the distinct spanning spiders of the graph. How many distinct spanning spiders does the graph in Figure 1 have?

[3.2] Write an ASP program `spidershortlegs.lp` that outputs a spanning spider with the shortest longest leg.