# Network Alert System - Java Tech Interview Assignment

## Assignment: Network Alert System

Imagine you're building a network alert system for a large-scale distributed service.
The system is modeled as a directed graph where each node represents a service or subsystem,
and edges represent alert propagation dependencies (i.e., if node A fails, it can trigger an alert in node B).
You are tasked with implementing core logic to determine how alerts spread and suggest optimization steps for containment.

### 1. Propagation Path Detection

Implement a method that finds the shortest path of alert propagation from a source system to a target system.
The method should return the ordered list of services affected along the path.
If there's no path between the two systems, return an empty list.
Expected Behavior: This feature is meant to detect how a failure at one service could potentially impact others downstream.

Method Signature:

List<String> findAlertPropagationPath(String source, String target);

### 2. Alert Impact Scope

Given a service that triggered an alert, implement a method to return a list of all services that could be affected
directly or indirectly by this alert. This effectively means performing a reachability analysis starting from the given node.

Method Signature:

List<String> getAffectedServices(String source);

## 3. (Bonus) Containment Suggestions

Suggest one or more edges (connections) that, if temporarily disabled, would minimize the alert spread from a given source.
Your implementation should prioritize cutting paths that isolate the largest number of downstream systems,
while cutting the fewest edges.

Method Signature:

List<Pair<String, String>> suggestContainmentEdges(String source);

## Provided Interface

```
public interface AlertNetwork {
   void addService(String service);
   void addDependency(String fromService, String toService); // Directed edge
   List<String> getDependencies(String service);

   List<String> findAlertPropagationPath(String source, String target);
   List<String> getAffectedServices(String source);
   List<Pair<String, String>> suggestContainmentEdges(String source); // Bonus
}
```

## Example Scenario

If you add the following dependencies:
addDependency("A", "B");
addDependency("B", "C");
addDependency("A", "D");
addDependency("D", "C");

- findAlertPropagationPath("A", "C") might return ["A", "B", "C"] (or ["A", "D", "C"])
- getAffectedServices("A") would return ["B", "C", "D"]
- suggestContainmentEdges("A") could return [("B", "C")] or [("D", "C")] depending on implementation

## Evaluation Criteria

- Correctness and completeness
- Use of appropriate graph algorithms (BFS, DFS, etc.)

- Code readability and documentation
- Efficiency (bonus)

# Reconstruct Ordered List

## Problem Description:

You are given a List<List<String>> called pairs, where each inner list contains exactly two strings:

- The first element represents a value
- The second element represents the name of the next value

Your task is to reconstruct the correct ordered list of values as a List<String>, starting from the first element in the chain and ending at the last (where the next value is null or missing in the data).

You can assume:
- There are no cycles
- The chain is continuous and contains no branching
- There is exactly one valid ordering

## Example Input:

```
List<List<String>> pairs = List.of(
    List.of("A", "B"),
    List.of("C", "D"),
    List.of("B", "C"),
    List.of("D", null)
);
```

## Expected Output:

```
List<String> ordered = List.of("A", "B", "C", "D");
```

## Instructions:

1. Implement the following method:

```
public static List<String> reconstructOrder(List<List<String>>
pairs) {
    // your code here
}
```

2. Add a few test cases in a main method to demonstrate your solution.

## Evaluation Criteria:

- - Correctness of the logic

- - Use of appropriate data structures
- - Code readability and structure
- - Handling of edge cases
- - Efficiency of your solution