1. **Reed-Solomon practice**: Consider the Reed-Solomon code from the lecture with parameters \( K = 1, N = 3, q = 5 \) and \( \alpha = 2 \).

   (a) Suppose we receive \( y^N = [2, 1, \bot] \). Fix the erasure error and decode the message.
   (b) Suppose we receive \( y^N = [1, 1, 2] \). Fix the error (if any) and decode the message.

2. **Reed-Solomon decoding algorithm**: Write out the Reed-Solomon decoding algorithm for the case of \( C \leq \lceil \frac{T}{2} \rceil \) errors at unknown locations in pseudocode.
   
   *Hint: Follow the procedure outlined in the lecture.*

3. **Message passing for counting vertices of graphs**: Recall that in the lecture we described a message passing algorithm for decentralized counting.

   (a) Describe how you could generalize the message passing algorithm to counting the number of vertices of graphs without cycles (these are called *trees*).
   (b) Work out explicitly what this algorithm would do for the graph below. Which node outputs the final answer?

   ![Graph](image)

   (c) Why does the algorithm fail in the presence of cycles? Find a concrete counterexample.

4. **Message passing for minimal cost paths**: Suppose we have a grid with two distinct nodes A and B and, as in the lecture, we can only walk steps down or to the right. Now we assume that each step has a cost associated to it, and we want to find the path from A to B with minimal total cost.

   (a) What is the separation property we can use?
   (b) Can you design a *forward* message passing algorithm that computes the minimal cost of a path between A and B?
   (c) Explain how you can use a backward pass to also obtain a minimal path (and not just the minimal cost).