Problem type 1:

Consider the problem of:

> *(See variants below)*

Please be concise/brief. We will grade kindly. There are lots of correct answers and we're just looking to make sure you understand how languages and problems are connected.

a. **BYH**

Checking whether (or not) a number is divisible by 4). You are given a binary number and need to output if this number is divisible by 4.

> **Solution:** Note that if a binary number is divisible by 4, then it must have two zeroes in the suffix. We enumerate all possible binary inputs and if they end in `00` we append a "| 1" at the end to denote that the number is divisible by four, "| 0" otherwise. ■

b. **BYF**

Summing two *unary* integers.

> **Solution:** We simply need a encodign scheme to convert each instance of the problem to a string. Could be done in a manner similar to what we saw in lecture:

$$L_{+unary} = \left\{ \begin{array}{ccc} +=, & +1=1, & +11=11,\ldots \\ 1+=1, & 1+1=11, & 1+11=111,\ldots \\ 11+=1, & 11+1=111, & 11+11=1111,\ldots \\ \vdots & \vdots & \vdots \\ n+=1....1\text{(n times)}, & \ldots, & n+11=1....1\text{(n+2 times)},\ldots \end{array} \right\} \quad (1)$$

■

c. **BYA**

The game of TicTacToe. You are given a completed tic-tac-toe board and you need to determine who won.

> **Solution:** So with the game of TicTacToe, first thing to notice is that there are a finite number of games. There are at most 3^9 possible boards (but remember not all boards are valid since the number o X's and O's must differ by at most 1. **We construct the language (L_{TTT}) by including a string for all the possible games of TicTacToe and their winners.** ■

d. **BYB**

Given a undirected weighted graph, the shortest path between 2 nodes s and t.

Solution: Again, this is just a issue of embedding a graph as a string which can be accomplished a multitude of ways. Big thing is to remember what a graph is, a set of nodes and a set of edges. So how do we encode a graph as a string? We simply list out the nodes and edges.

Each string in a language represents an *instance* of the problem. In this case, the problem input is the weighted, undirected, graph and the output is the shortest path. Knowing this, we can formulate a string embedding such as:

$$\langle < listofnodes > | < listofedges > | < shortestpath > \rangle$$

■

Problem type 2:

Give the recursive definition for the following language:

(See variants below)

Assume $\Sigma = \{\mathtt{0}, \mathtt{1}\}$

a. **BYC**

A language that contains all strings.

Solution: Base case: $\varepsilon \in L_a$

- $w = \mathtt{0}x$ for some $x \in L_a$, is in L_a
- $w = \mathtt{1}x$ for some $x \in L_a$, is in L_a

■

b. **BYE**

A language which holds all the strings containing the substring **000**.

Solution:
- **Base case:** $\mathtt{000} \in L_a$
- $w = \mathtt{0}x$ for some $x \in L_a$, is in L_a
- $w = \mathtt{1}x$ for some $x \in L_a$, is in L_a
- $w = x\mathtt{0}$ for some $x \in L_a$, is in L_a
- $w = x\mathtt{1}$ for some $x \in L_a$, is in L_a

■

c. **BYD**

L_A that contains all palindrome strings using some arbitrary alphabet Σ.

Solution:
- $w = \varepsilon$, or
- $w = \texttt{a}$ for some symbol $a \in \Sigma$, or
- $w = \texttt{a}x\texttt{a}$ for some symbol $\texttt{a} \in \Sigma$ and some palindrome $x \in \Sigma^*$

■

d. **BYG**

A language which holds all the strings containing the substring **`000`**.

Solution: See above ■