Let Σ = { 0, 1, ||, Δ, ⨆ } for both questions, though you may add to this if you need to. The Δ represents the leftmost marker on the tape. The ⨆ represents the rightmost marker on the tape. The 0 and 1 will be used to represent binary integers. Each problem assumes a pair of binary integers are given on the tape, and the || will be used to separate one binary integer from another.

For example, for each question will assume the tape format Δ m || n ⨆ where both m and n are binary integers. Example initial tape conditions may be Δ00101110||00011111 ⨆ and Δ111||001000⨆ (representing the numeric pairs 46 || 31 and 7 || 8 - if my math is right).

Provide a description of the tape operations to solve the following problems. Use the examples in the Cornell notes as a guide to the level of detail required.

1. (15 pts) Problem 1: Given two binary integers, m and n, of equal length, return "yes" if m=n and "no" otherwise.

NOTE: Problem says you are given two binary integers of EQUAL LENGTH so I will work under the assumption that both will have an equivalent number of binary digits. One benefit of this setup/assumption is that it means that every single digit between each pair must be the same for it to return true.

A very simplified explanation of how this Turing machine would work is…

* Start at LHS and start state, traverse right to very first digit following the Δ (leftmost bit in integer m).
* Remove that first bit and traverse to the first bit following the “||” (aka the leftmost bit in integer n) and remove that bit and compare it to the other removed bit from integer m.
  + If they are the same, return left to Δ and repeat process until every bit has been removed and all that remains is “Δ ||⨆” as the final string, from which “yes” will be returned.
  + If they are not the same, return “no” and end process.

1. (15 pts) Problem 2: Given two binary integers, m and n, that may not be of equal length, return "yes" if m > n, and "no" otherwise.

A very simplified explanation of how this Turing machine would work is…

* Start at LHS and start state, traverse right until you reach “||” then travel back left once and (as long as that symbol isn’t Δ) read and remove it.
  + If the value that is read is Δ, don’t remove anything, just enter state that assumes that the read m bit is empty.
* Following previous step, travel right until you reach ⨆ and then travel back left once and (as long as that symbol isn’t “||”) read and remove it.
  + If the value that is read is “||”, don’t remove anything, just enter state that assumes that the read n bit is empty.
* Following previous step, compare the two read values and if the read m bit is 1 and the read n bit is 0 or empty enter the state branch that reflects m>n and return left to Δ and repeat the process.
  + If the read m bit is 1 and the n bit is 1 (or m is 0 and n is 0): if are in the branch that reflects m > n return left to Δ and to beginning of m>n branch and repeat process. If not in m>n branch, stay in m<=n branch and return left to Δ and to beginning of m<=n branch and repeat process.
  + If n bit is 1 and m bit is 0 or empty, enter the state branch that reflects m<=n and return left to Δ and repeat the process.
* While comparing, if both the n bit and m bit are empty, jump to end of current branch and return “yes” or “no” based on if in the m>n branch or the m<=n branch respectively and terminate the process.

A MORE ADVANCED EXPLANATION OF HOW THE TURING MACHINE IN PROBLEM 1 MAY WORK…

NOTE: Problem says you are given two binary integers of EQUAL LENGTH so I will work under the assumption that both will have an equivalent number of binary digits. One benefit of this setup/assumption is that it means that every single digit between each pair must be the same for it to return true.

Basically, it will probably have main states Q = {s, l, q0, q1, r0, r1 } start from state “s” which basically does nothing but transition to state “l” as it reads Δ and begins moving right.

From state “l” it will remove the first bit it reads following the Δ, where based on if it reads a 0 or a 1 it will transition to state q0 or q1 respectively. If state “l” reads a “Δ ||⨆” as the final string it will just be returning “yes” and ending the process.

From state q0 or q1 it will then continue going right not altering any bits until it reads a “||” where the following bit will determine where it goes. It will read and remove this first bit it detects after the “||”…

* If it is in q0…
  + And reads a 0: after removing that bit it goes to r0 (meaning it has determined that m and n share a 0 for this digit), where in this state the machine will transition back to the far left without altering any bits until it reaches Δ where it will then transition to state “l” and repeat the process.
  + And reads a 1: it will simply return “no” and end the process.
* If it is in q1…
  + And reads a 0: it will simply return “no” and end the process.
  + And reads a 1: after removing that bit it goes to r1 (meaning it has determined that m and n share a 1 for this digit), where in this state the machine will transition back to the far left without altering any bits until it reaches Δ where it will then transition to state “l” and repeat the process.

This process will continue until it either returns a “no” by detecting an unmatched digit pair or by returning a “yes” when, from state “l”, the first symbol it reads is a “||” which would imply that it has cleared all bits as matching.