Problem 6.1. (15 points)
Consider the following language.

A is the set of all words over the alphabet $\Sigma = \{\mathtt{0}, \mathtt{1}, \#\}$ that have the form

$$u_1 u_2 \dots u_k \# v_1 v_2 \dots v_m, \quad \text{where } k, m \geq 1,$$

for binary numbers $u = u_1 \dots u_k$ and $v = v_1 \dots v_m$ such that $u \leq v$.

Prove that A is recursive. For this purpose, give both a high-level description and a state-transition diagram of a Turing decider for A.

Hint: The machine should look at v and u bit by bit starting from the least significant bits. The machine should figure out cases when it has reached the leftmost position of its tape.

Solution:
The TM uses the tape alphabet $\Gamma = \Sigma \cup \{\mathsf{x}, \mathsf{y}, \sqcup\}$. We can declare that u is smaller or equal than v if $u = v$ or the most significant bits that differs in u and v are such that the bit of v is larger than the corresponding bit of u.
The machine looks at v and u bit by bit starting from the least significant ones, compares each bit v_i with u_i, and remembers (in its state) at each point whether the part of u considered so far is indeed smaller or equal to the part of v considered so far. The reason why we check bits starting from the least significant ones is to make sure that we are looking at the bits corresponding to the same power of two in u and v (We know that the rightmost bit of u and v is the least significant one, but we do not know whether the leftmost bit of u and v have the same significance.) To remember which bits have already been checked, we replace checked bits by x in u and v. Since u and v may have different length, if we notice that all bits of u have already been checked while looking for the i-th bit of u, we do the same as if this bit had a value of 0. Note that for noticing whether we are at the leftmost position of the tape, we use the fact that the TM will stay at the same position if the transition specifies to move left (L) and it is already in the leftmost tape cell. That is why, when moving to the left in the u part, we replace x's by y's; symmetrically, we replace y's by x's when moving right in the u part. If we eventually have a y on the tape while moving left, it means that we are at the beginning of the tape. The TM accepts if, when all bits of v have been checked, the machine knows that the inequality holds so far, and all non-checked bits of u are 0's.
Considering this algorithm, we get the TM in Figure 1. Note that Σ_{x} denotes $\Sigma \cup \{\mathsf{x}\}$. The left-hand side (states framed by single continuous line) of the machine represents situations where so far, the inequality holds; the right-hand side (states framed by dotted continuous line), situations where where so far, the inequality does not hold.

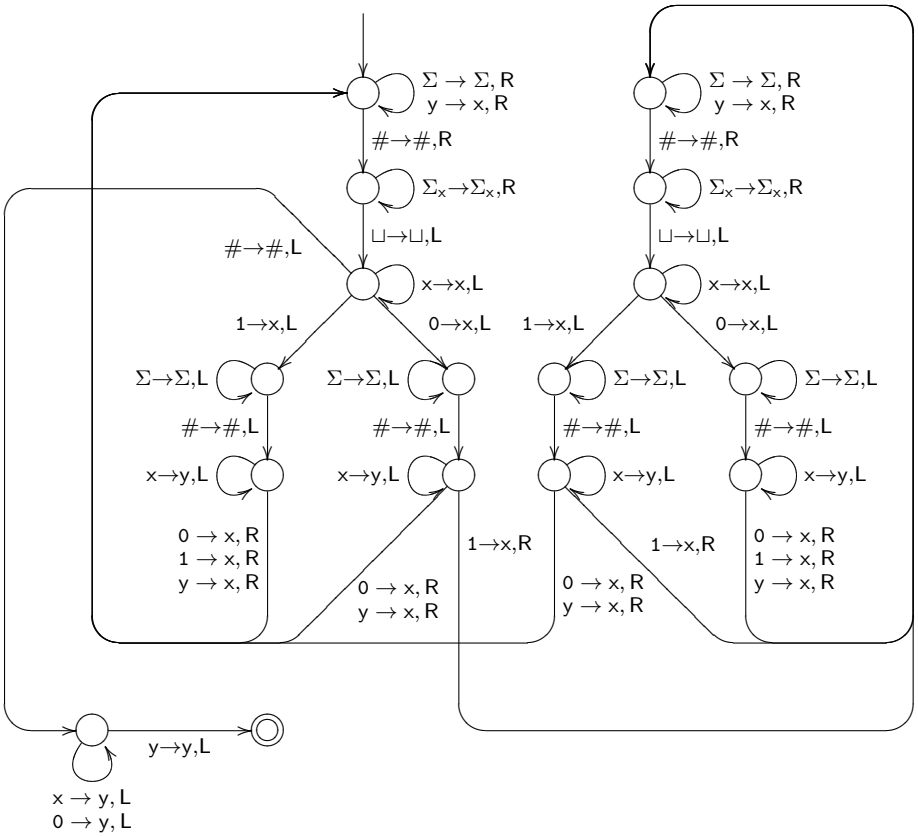

Figure 1: Turing decider for A.

Problem 6.2. (15 points) Assume that C and D are r.e. languages, and $C \cap D$ and $C \cup D$ are both recursive. Prove that both C and D are recursive. (Given Turing recognizers for C and D, and Turing deciders for $C \cap D$ and $C \cup D$, construct Turing deciders for C and D. Give high-level descriptions for the constructed Turing deciders for C and D.)

Solution:
By assumption, we know that there exists:

- TM M_C and M_D such that $L(M_C) = C$ and $L(M_D) = D$,
- TD $M_{C \cap D}$ and $M_{C \cup D}$ such that $L(M_{C \cap D}) = C \cap D$ and $L(M_{C \cup D}) = C \cup D$.

We will give a decider D_C for C. The decider for D is symmetrical.
Suppose D_C is given the string x. D_C would behave as follows:

1. Query $M_{C \cup D}$ whether it accepts x or not. If it rejects, REJECT. If it accepts, proceed.
2. Query $M_{C \cap D}$ whether it accepts x or not. If it does, ACCEPT. Else proceed.
3. At this stage we know that either C or D contains x, but not both. We just have to decide which one. Therefore, simulate M_C, M_D on x for $i = 1, 2, \ldots$ steps, until one of the TM accepts, which we know must happen at some point in time. If the accepting recognizer was M_C, ACCEPT, else REJECT.