1. Consider a Turing machine over the input alphabet \{0, 1\}, with set of states \{s_0, s_1, s_2, t, r\}, initial state \(s_0\), accept state \(t\), reject state \(r\), and with transition function given by:

<table>
<thead>
<tr>
<th></th>
<th>(\downarrow)</th>
<th>0</th>
<th>1</th>
<th>(\sqcup)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_0)</td>
<td>((s_0, \downarrow, R))</td>
<td>((s_1, 0, L))</td>
<td>((s_1, 1, L))</td>
<td>((s_0, \sqcup, R))</td>
</tr>
<tr>
<td>(s_1)</td>
<td>((s_1, \downarrow, R))</td>
<td>((s_2, 1, R))</td>
<td>((s_1, 0, R))</td>
<td>((s_2, 1, R))</td>
</tr>
<tr>
<td>(s_2)</td>
<td>((r, \downarrow, R))</td>
<td>((s_2, 0, R))</td>
<td>((s_2, 1, R))</td>
<td>((t, \sqcup, L))</td>
</tr>
</tbody>
</table>

(a) What is the outcome of executing \(M\) on input \(\downarrow 110\) ?
(b) How many times does the machine reach a configuration where the head points to the left endmarker (including the initial configuration), when started on the input \(\downarrow 1111110\) ?
(c) Is this Turing machine total?
(d) What is the language accepted by this Turing machine?

**Solution.** For any non-empty input \(x_1 \ldots x_n\) (\(n \geq 1\)), thought of as the binary representation of a natural number \(n\), with \(x_1\) the least significant and \(x_n\) the most significant digit, the machine will eventually replace this input by the binary representation of \(n + 1\). On an empty input, the machine loops, as the tape head keeps moving to the right infinitely. So:

(a) On \(\downarrow 110\), the machine accepts with 001 on its tape.
(b) On \(\downarrow 1111110\) (and indeed any non-empty input), the head points to the left endmarker at the very beginning, and then again after two transitions. Once state \(s_1\) is reached, the tape head always moves to the right, and therefore the left endmarker will not be seen again.
(c) The machine is not total as it loops on the empty input.
(d) The machine accepts \(\{0, 1\}^* \setminus \{\epsilon\}\).

2. Describe a TM that accepts the set \(\{ww \mid w \in \{a, b\}^*\}\). You do not have to mention all the transitions, but describe the design in enough detail so that if someone paid you enough, you could.

**Solution.** First we scan the tape and put a \(\downarrow\) at the end of the input.

\(\downarrow abbaabba\ldots\)

Next we scan back and mark the first symbol with a \(\acute{}\) accent.

\(\downarrow abbaabb\acute{a}\ldots\)
Now we scan again from left to right and mark the first symbol with a \( \hat{\_} \) accent.

\[ \hat{\_} \text{abb}a \hat{\_} \text{abb} \hat{\_} \text{a} \]

We continue this process until we have accented all the symbols.

\[ \hat{\_} \text{abb}a \hat{\_} \text{abb} \hat{\_} \text{a} \]

By doing so we have found the centre of the string. Now we will scan from left to right, delete the first \( \hat{\_} \) marked symbol, remembering in which symbol we have deleted – (here \( \hat{\_} \)):

\[ \hat{\_} \text{ab} \hat{\_} \text{a} \hat{\_} \text{b} \hat{\_} \text{b} \]

and continue scanning until we come to the first \( \hat{\_} \) accented symbol – if the symbol agrees with the one we have just deleted then we delete it and return to the start of the string.

\[ \hat{\_} \text{ab} \hat{\_} \text{a} \hat{\_} \text{b} \hat{\_} \text{b} \]

If the symbol is not the same, or if there is no \( \hat{\_} \) accented symbol, we reject. Now we continue doing the last two steps, deleting a \( \hat{\_} \) symbol and an \( \hat{\_} \) symbol in pairs. We stop and accept if we end up with all the symbols deleted.

\[ \ldots \]

3. Prove that:

(i) recursively enumerable sets are closed under union and intersection;

**Solution.** Suppose that we have r.e. sets \( A \) and \( B \). Then there are TMs \( M \) and \( N \) so that \( L(M) = A \) and \( L(N) = B \). We need to show that \( A \cup B \) is r.e. so we will construct a TM \( K \) so that \( L(K) = A \cup B \).

In Lecture 14 we discussed how to construct a TM that simulates the running of two TMs. The construction works by encoding the two tapes of \( M \) and \( N \) on one tape and then simulating one move of \( M \), followed by one move of \( N \), followed by one move of \( M \), and so on.

So let \( K \) on \( x \) simulate the running of both \( M \) and \( N \) on \( x \), accepting if either accepts and rejecting if both reject. Then clearly \( L(K) = M \cup N \). It is easy to convert the construction to accept the intersection – we accept precisely when both \( M \) and \( N \) accept and reject if either rejects.

(ii) recursive sets are closed under union and intersection.

**Solution.** Here we can repeat the same solution as for part (i), noting that if \( M \) and \( N \) are total TMs then also \( K \) will be total.

Alternatively, if \( A \) and \( B \) are recursive then also \( \neg A \) and \( \neg B \) are recursive (Lecture 14) and so \( A, B, \neg A, \neg B \) are all r.e. Now we know from (i) that \( A \cup B \) is r.e. but also \( \neg (A \cup B) = \neg A \cap \neg B \) is r.e. Now \( A \cup B \) and \( \neg (A \cup B) \) are both r.e. implies that \( A \cup B \) is recursive (Lecture 14).

Similarly \( \neg (A \cap B) = \neg A \cup \neg B \) is r.e. so also \( A \cap B \) is recursive.