Problem 1: Define the following words, phrases and symbols.

1. Turing Machine

2. \((Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})\)
   - \(Q\): states
   - \(\Sigma\): input alphabet, not including the blank symbol
   - \(\Gamma\): tape alphabet, includes blank and \(\Sigma\) and possibly other symbols
   - \(\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}\)
   - \(q_0\): start state
   - \(q_{\text{accept}}\): accept state
   - \(q_{\text{reject}}\): reject state, \(q_{\text{reject}} \neq q_{\text{accept}}\)

3. (Turing) Decidable Language
   Turing machine halts on all inputs (Sipser p.142)

4. (Turing) Recognizable Language
   Turing machine accepts strings in the language, halts loops on strings not in the language (Sipser p.142)

5. (Recursively) Enumerable Language
   There exists an enumerator that enumerates (prints all the strings in) the language (Sipser p.152)

6. Looping / Not Halting
   Turing machine continues on forever without halting

Problem 2: Mark each of the following statements either true or false.

1. A Turing machine has a single start state, but may have many accept states.
   False. It has a single accept state \(q_{\text{accept}}\).

2. It is possible to make a Turing machine with only one state.
   Yes. One or both of \(q_{\text{accept}}\) and \(q_{\text{reject}}\) must be \(\emptyset\).

3. A Turing machine halts when its head reaches the end of its input.
   False. It can do anything it wants when it reaches the end of its input, such as go back to the beginning and continue processing it.

4. All decidable languages are regular languages.
   False. Consider \(O^n 1^n\) or many other examples.

5. A nondeterministic TM with \(k\)-heads can recognize more languages than a deterministic TM with \(k\)-tapes.
   False. NTMs and DTMs are equivalent, and TMs with different numbers of tapes and heads are equivalent. See Sipser for proofs.

6. A Turing machine might not halt on a finite input string.
   True. It can loop forever regardless of the input string.
7. A language $L$ can be both co-decidable and undecidable. False. If $L$ is co-decidable by definition $\overline{L}$ is decidable. A TM deciding $L$ can be built by swapping $q_{\text{accept}}$ and $q_{\text{reject}}$ in the decider for $\overline{L}$, so $L$ must also be decidable.

**Problem 3:** Describe the operation of a basic Turing machine that recognizes the language $L = \{ww^R : w \in \{0, 1\}^*\}$. Explain how the Turing machine head(s) move and mark the tape(s), without listing the specific details of each transition. (This is what Sipser calls an “implementation description” on p. 157.)

How would you recognize the language $\{0^n10^n2^n10^n3^n : n \geq 1\}$ using a 3-headed, single tape TM?

**Solution 3:**

$M = \text{"On input string } x:\"

1. If the input is empty accept. Otherwise mark the beginning of the input.

2. Scan the input from left to right to verify that all symbols are either 0 or 1, and determine if there are an even number of symbols, reject if not. Mark the end of the input and move back to the beginning of the input.

3. Store the first symbol in the input and cross it off. Move the head to the end of the input and check if the last symbol matches the stored symbol, and reject if not. If it does match cross it off.

4. Move to the left. If the symbol is crossed off then all the symbols have matched and been crossed off so accept. Otherwise continue moving left until a crossed off symbol is reached. Move right. Store the symbol and move right until a crossed off symbol is read. Move left. Check if the symbol matches the stored symbol and reject if not. If it does match cross it off.

5. Repeat step 4 until either rejecting or crossing off all the input and accepting. ”

For $L = \{0^n10^n2^n0^n3^n : n \geq 1\}$:

$M = \text{"On input string } x:\"

1. Scan the input from left to right to determine whether it is a member of $0^+10^+10^+$ and reject if is isn’t.

2. Position the first head on the first 0 of the input string. Position the second head on the 0 after the first 1. Position the third head on the 0 after the second 1.

3. Cross off the 0 under each head. Move the second head to the right. If the symbol is a 0 cross it off, otherwise reject. Move the third head to the right. If the symbol is a 0 cross it off, move right and check for another 0. If there is another 0 cross it off, otherwise reject.

4. Move each head to the right. If each head is on a 0 repeat Step 3. If the first and second heads are on 1s and the third head is on a blank then accept. Otherwise reject.”

**Problem 4:** Show that the class of decidable languages is closed under concatenation.

**Solution 4:** Let $L_1$ and $L_2$ be two decidable languages, then by definition there are TMs $M_1$ and $M_2$ that decide these languages. We construct a 3-tape TM

$M = \text{"On input } x:\"

1. Nondeterministically split the input string into two parts $x = yz$ and copy $y$ to the second tape and $z$ to the third tape.

2. Simulate $M_1$ on $y$ on the second tape. If $M_1$ rejects then reject, otherwise $M_1$ accepts so continue

3. Simulate $M_2$ on $z$ on the third tape. If $M_2$ accepts then accept, else $M_2$ rejects, so reject”

4: Solution Sketches-2
If \( x \in L_1 \circ L_2 \) then one of the nondeterministic paths will split it such that both \( M_1 \) and \( M_2 \) will accept, causing \( M \) to accept. As well, if \( x \notin L_1 \circ L_2 \) then no matter how it is split it must be the case that either \( L_1 \) or \( L_2 \) will reject, causing \( M \) to reject. A 3-tape nondeterministic TM is equivalent to a single-tape deterministic TM, therefore we can construct a single-tape deterministic decider for the concatenation of \( L_1 \) and \( L_2 \).

**Problem 5:** [A Different Turing Machine Model] A Turing Machine (TM) with doubly infinite tape is similar to an ordinary TM except that its tape is infinite to the right as well as left. The tape is initially filled with blanks except for the portion that contains the input. Computation is defined as usual except that the head never encounters an end to the tape as it moves leftward. Show that this type of TM recognizes the class of Turing-recognizable languages.

**Solution 5:** We show that using a regular TM we can simulate a TM with a double infinite tape (2-way TM).

First for an arbitrary configuration of a 2-way TM \( M_2 \) we show how it can be represented by a configuration of a regular TM \( M_1 \). For any configuration, past some point in each direction the tape must be blank:

\[
\ldots \quad -2 \quad -1 \quad 0 \quad 1 \quad 2 \quad \ldots
\]

\[
\ldots \quad \text{t a p e} \quad \text{c o n t e n t} \quad \text{s} \quad \ldots \quad \ldots
\]

We can represent this by folding the tape over so that each tape symbol of \( M_1 \) is a pair of tape symbols of \( M_2 \).

\[
\begin{array}{cccccc}
0 & 1 & 2 & \ldots \\
-1 & -2 & -3 & -4 & -5 & \ldots \\
\end{array}
\]

To represent an arbitrary configuration we also need to represent the location of \( M_2 \)’s head. \( M_1 \)’s head can point to the position where \( M_2 \)’s head is, but it could be on the upper track or the lower track, so we add states to the Finite State Control of \( M_1 \) to store which track the head of \( M_2 \) is on.

The input is presented to \( M_1 \) normally, so we need to show how to convert this initial configuration into a representation of the initial configuration of \( M_2 \) in which the symbols are all on the top track. We do this by converting each symbol to a pair \((a, \_\)\) (this works if we number \( M_1 \)’s tape squares from 0).

\( M_1 \) can then simulate \( M_2 \) by modifying its behavior based on which track \( M_2 \)’s head is on. We make sure that when reading symbols we only pay attention to the symbol on the track that \( M_2 \)’s head is on, and likewise when writing symbols we only alter the symbol on that track. Also, when \( M_2 \)’s head is on the bottom track a move left by \( M_2 \) is actually accomplished by a move right and vice versa.

**Problem 6:** Robustness of the Turing Machine model

Consider a Turing machine model that uses a 2-dimensional tape, corresponding to the upper right quadrant of the plane. The head of such a Turing machine could move to the right, left, up or down. Sketch a proof that such a model does not add extra computing power; that is, the class of languages recognized by such Turing machines is the same as the class recognized by basic Turing machines.

**Solution 6:** The key idea is to notice that at any point of time, the TM has looked at only a finite region of the two-dimensional tape.

Say the input of the machine is in the first row of the tape, initially the tape head is at the square \((0, 0)\) and all the other tape squares are blank.

Simulate the 2D tape with a 1D tape as follows:

1. The rows of the 2D tape starting from the “bottom-most” row are written in the 1D tape next to each other, as in the figure below. Note that this is indeed possible, because at any point of time, the TM has looked at only a finite number of rows, and a finite number of tape squares in each row.

2. When the machine tries to move right from a square in the 2D tape, we can simulate it in the 1D tape trivially, except when it tries to move right from the rightmost square in a row. In that case, we “make room” for the new tape square by moving all the symbols to the right. For instance, when the machine
tries to move right after reading $x_7$ in the figure, we simulate the effect in the 1D tape as follows: We move all the symbols starting from the $\$$ right after $x_7$ in the 1D tape, one square to the right. In the newly vacant square, we write a blank, and continue simulating.

3. When the machine tries to move left from the leftmost square in a row, we do nothing (*The machine hits the wall*). Otherwise, we can trivially simulate it in the 1D tape.

4. When the machine tries to move up, we move left until we reach a $\$$ symbol to determine which column we are in. Then we move right beyond the next $\$$ and into the corresponding column one row up. (To count, we might use an auxiliary tape and later use the 2-tape TM to 1-tape TM conversion). If there is not enough room in the next row, we make room by moving all the tape squares to the right, just as in step 2.

5. The machine tries to move down. We proceed exactly as in Step 4, except that now, we move one row down. In case we are in the bottommost row, we do nothing. (*Again, the machine hits the wall*).

---

The configuration of the 2D tape at some point of time

The representation of the 2D configuration in the 1D tape