Problem 1: In all parts of this question, the alphabet $\Sigma$ is $\{0, 1\}$.

1. True or False: If $w$ is any string, then $w^R$ denotes the reverse of $w$. If $L$ is any language, then $L^R$ denotes the language consisting of the reverses of all strings in $L$. If $L$ is any language, then the language $LL^R$ must be equal to $\{ww^R : w \in L\}$.

2. True or False: We say that one string $w$ is a subsequence of another string $x$ if $w$ can be obtained from $x$ by deleting zero or more symbols of $x$. For example, 1101 is a subsequence of 011001 but is not a subsequence of 101110.

Define $\text{Subseq}(L)$, for a language $L$, to be the set of strings $w$ that are subsequences of strings in $L$. If $L$ is a regular language, then $\text{Subseq}(L)$ must be regular.

3. True or False: Every NFA can be converted to an equivalent one that has a single accepting state.

4. True or False: For every three regular expressions, $R$, $S$, and $T$, the languages denoted by $(R \cup S)^*T^*$ and $R^*T^* \cup S^*T^*$ are the same.

5. True or False: A DFA with $n$ states that accepts an infinite language must accept at least two distinct strings $w$ and $x$ such that $n < |w| \leq 3n$ and $n < |x| \leq 3n$.

6. True or False: If $L_1$, $L_2$, and $L_3$ are all Turing-recognizable, then $L_1 \cap (L_2 \cup L_3)$ must be Turing-recognizable.

7. True or False: If $L_1$, $L_2$, and $L_3$ are all Turing-recognizable, then $\overline{L_1} \cap (L_2 \cup L_3)$ must be Turing-recognizable.

8. True or False: If $L_1 \leq_m L_2$ and $L_1$ is Turing-recognizable, then $L_2$ must also be Turing-recognizable.

9. True or False: Rice’s Theorem implies that $\{\langle M \rangle \mid L(M) \text{ is recognized by a Turing machine with at most 10 states and at most 10 tape symbols} \}$ is undecidable.

10. True or False: Suppose that language $L$ is recognized by a Turing machine variant that works on a 3-dimensional “tape” that is infinite in the positive direction in all three dimensions. Then $L$ is recognized by an ordinary basic Turing machine.
**Problem 2:** Consider the following NFA:

![NFA Diagram]

1. Convert this NFA into an equivalent DFA using the procedure we studied in class. Your answer should be the state diagram of a DFA. Your diagram should include only the states that are reachable from the start state. (Note: There are no more than six states in the resulting DFA). Please label your states in some meaningful way.

2. What language is recognized by your DFA? Your answer may be either a regular expression or an explicit description of the set.

**Problem 3:** Regular expressions are defined using three operators: union, concatenation, and star. Suppose we define “Extended Regular Expressions” in the same way as regular expressions, with the addition of the set complement operator. For example, \((0^*1 \cup 01^*)\) is an extended regular expression, which happens to denote the set consisting of the single string 01.

Show that adding the complement operator does not extend the power of ordinary regular expressions. Do this by describing a procedure (something that can be implemented using a program) that, given an extended regular expression \(\alpha\), produces an ordinary regular expression \(\beta\) that represents the same language. You may use procedures described in class and in Sipser’s book without saying how they work, e.g., you may say things like “convert the NFA to a DFA”. The description of your procedure should be concise, but the procedure need not be the most efficient one possible.

**Problem 4:** Prove that the following language \(L\) over the alphabet \(\{0, 1\}\) are not regular:

1. \(L = \{0^m1^n \mid m \neq n\}\)
2. \(L = \{wtw \mid w, t \in \{0, 1\}^+\}\)
Problem 5:
Consider the following formal description of a Turing Machine $M$, where $Q = \{q_0, q_1, q_2, q_3, q_4, q_{\text{reject}}, q_{\text{accept}}\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, X, \sqcup\}$. Assume that any unspecified transitions go to $q_{\text{reject}}$.

1. Write out the accepting computation history of $M$ on input 00.
2. Describe in a few words the behavior of $M$ on input 01.
3. What language does $M$ recognize?
4. Recall the mapping reduction done in class (and in the textbook) from $A_{TM}$ to the Modified Post Correspondence Problem, and consider what that mapping would produce for Turing Machine $M$. Specifically, what tiles are added to the instance of MPCP produced from $M$, to represent all the moves (both left and right) out of state $q_1$?

Problem 6: Let $FIN = \{\langle M \rangle | M$ accepts only a finite number of strings$\}$.

1. Does Rice’s Theorem apply to $FIN$? Why or why not?
2. Prove the following two results about $FIN$. You may use any results proved in class or in Sipser’s book, but if you do, then cite the results explicitly.
   (a) Use mapping reducibility to prove that $FIN$ is not Turing-recognizable.
   (b) $\overline{FIN}$ is not Turing-recognizable, that is, $FIN$ is not co-recognizable. (Hint: Hard. Use mapping reducibility.)
Problem 7:

Define a Turing machine $M$ to be almost-minimal if there does not exist another Turing machine $M'$ such that $L(M') = L(M)$ and $|\langle M' \rangle| < \frac{1}{6} |\langle M \rangle|$. That is, there is no machine that recognizes the same language and has an encoding whose size is less than one-sixth of the size of $\langle M \rangle$.

Prove carefully that there is no enumerator that outputs a set $S$ consisting of an infinite number of almost-minimal machine descriptions. (Hint: Proceed by contradiction and use the Recursion Theorem.)

Problem 8: In your summer internship at SmartCompilers, Inc, your boss wants you to implement a new tool for their C++ development environment: a program to test whether an arbitrary C++ program can ever access an array element that is outside of the array’s declared bounds.

That is, let SAFE be defined as the language $P$ — $P$ is a C++ program and $P$ does not access any array element outside of the array bounds; your boss is asking for a program that decides SAFE.

Give a formal argument for why this task is impossible—that is, show that SAFE is undecidable.