You have 90 minutes to complete this exam. You may state without proof any fact taught in class or assigned as homework.

1. Give a simple verbal description of the language recognized by the following DFA.

```
  1  0  1  0
  0  1  1  0
```

**Solution.** Binary strings in which the first and last symbols differ.

2. Draw NFAs for the following languages, taking full advantage of nondeterminism:

   a. binary strings that contain at least two 0s and end with a 1;
   b. binary strings that contain 0101 or 11101 or both as a substring.

**Solution.**

```
  1  1  0, 1
  0  0  1  
```

a. 

```
  0, 1
```

b.
3. Prove that the following languages are regular:

(2 pts) a. binary strings in which the number of 0s and the number of 1s are equal modulo 2016;
(2 pts) b. binary strings in which every 0 is immediately preceded and immediately followed by a 1;
(2 pts) c. binary strings of length 8 that read the same backward as forward.

Solution.

a. A DFA for this language just counts the number of 0s and the number of 1s modulo 2016, accepting if the numbers match. Formally, the DFA is

\[ (\{0, 1, \ldots, 2015\} \times \{0, 1, \ldots, 2015\}, \{0, 1\}, \delta, (0, 0), \{(0, 0), (1, 1), \ldots, (2015, 2015)\}), \]

where

\[ \delta((i, j), \sigma) = \begin{cases} ((i + 1) \mod 2016, j) & \text{if } \sigma = 0, \\ (i, (j + 1) \mod 2016) & \text{if } \sigma = 1. \end{cases} \]

b. This language is recognized by the following DFA:

![DFA Diagram]

c. This language contains a finite number of strings and is therefore regular.
Describe an algorithm that takes as input two DFAs and determines whether there is a string that they both accept. Your algorithm must run in finite time.

Solution. Let \( D_1 \) and \( D_2 \) be the input DFAs. Use the Cartesian product construction from class to obtain a DFA \( D \) that recognizes the intersection of the languages of \( D_1 \) and \( D_2 \). Then, check to see if \( D \) has a path from the start state to an accept state. This second step can be done either by trying out all candidate paths in a brute force manner, or by using an efficient graph algorithm that you have encountered in CS 180, such as depth- or breadth-first search.

For a language \( L \), let \( \text{trap}(L) \) denote the set of strings \( v \) such that no string in \( L \) starts with \( v \). Prove that \( \text{trap}(L) \) is regular whenever \( L \) is regular.

Solution. Let \( (Q, \Sigma, \delta, q_0, F) \) be a DFA for \( L \). Then \( \text{trap}(L) \) is recognized by \( (Q, \Sigma, \delta, q_0, F') \), where \( F' \subseteq Q \) is the set of all states from which it is impossible to reach any state in \( F \).

Alternate solution. Observe that \( \text{trap}(L) = \overline{\text{prefix}(L)} \). Since regular languages are closed under the prefix and complement operations, it follows that \( \text{trap}(L) \) is regular.
For a language $L$ over a given alphabet $\Sigma$, define $\text{swap}(L)$ to be the set of strings that can be obtained from a string in $L$ by swapping two characters in it. Formally, $\text{swap}(L)$ is the set of all strings of the form $u\sigma v\tau w$ such that $u\tau v\sigma w \in L$, where $u, v, w$ denote arbitrary strings and $\sigma, \tau$ denote alphabet symbols. Prove that regular languages are closed under the swap operation.

**Solution.** Let $(Q, \Sigma, \delta, q_0, F)$ be a DFA for the original language $L$. An NFA for $\text{swap}(L)$ is as follows.

For a string $v$, a *subsequence* of $v$ is either $v$ itself or any string obtained by deleting one or more characters in $v$. For example, the subsequences of $123$ are $\varepsilon, 1, 2, 3, 12, 13, 23, 123$. Prove that for any regular language $L$, the set of all subsequences of strings in $L$ is regular.

**Solution.** Starting with a DFA for $L$, supplement the label of every arrow to contain “$\varepsilon$”. The resulting NFA accepts exactly those strings that are subsequences of strings in $L$. 