Problem 1 True/False (10 points, 1 point each) Answer True or False. The languages are over the alphabet $\Sigma = \{0, 1\}$. No justification required.

	Question	True	False
1	The CFG with the rules $S \to SS \mid Sb \mid a \mid ab$ is ambiguous.	✓	
2	If A is regular and $B \subseteq A$, then B is regular.		✓
3	If A reduces to B and B is undecidable, then A is undecidable.		✓
4	$(0+1)^* = 0^* + 1^*$.		✓
5	The regular expression $(0 + 11 + 10)^*1^*$ generates all binary strings	✓	
6	If a language L is recognized by an NFA, then $\bar{L}$ is recognized by an NFA.	✓	
7	If A is regular and B is context-free, then $A \cup B$ may not be context-free.		✓
8	If L is undecidable, then L is infinite.	✓	
9	L is decidable if and only if its complement $\bar{L}$ is undecidable.		✓
10	If A is regular and $A \cup B$ is regular, then B is regular		✓

Problem 2 Short answer questions (16 points, 2 points each)

1) Is it true that, for all languages L_1 and L_2, we have $(L_1^* \cap L_2^*)^* = (L_1 \cap L_2)^*$? Give a short proof if you believe the statement is true or give a counter-example if you believe the statement is false.

False. Take $L_1 = \{0\}$, $L_2 = \{00\}$ (over $\Sigma = \{0,1\}$)
then $L_1^* \cap L_2^* = (00)^*$, so $(L_1^* \cap L_2^*)^* = (00)^*$.
However $L_1 \cap L_2 = \emptyset$, so $(L_1 \cap L_2)^* = \emptyset$.

2) Give a regular language over $\Sigma = \{a, b\}$ with a one-state NFA but no one-state DFA. Briefly justify your answer.

$L = a^*$
A one-state NFA for L : 

A DFA for L must read b in a different state from the one it accepts the strings in L, thus it must have at least two states.

3) Give a regular expression for the language consisting of strings over $\{0,1\}$ in which the number of 0s and the number of 1s are either both even or both odd.

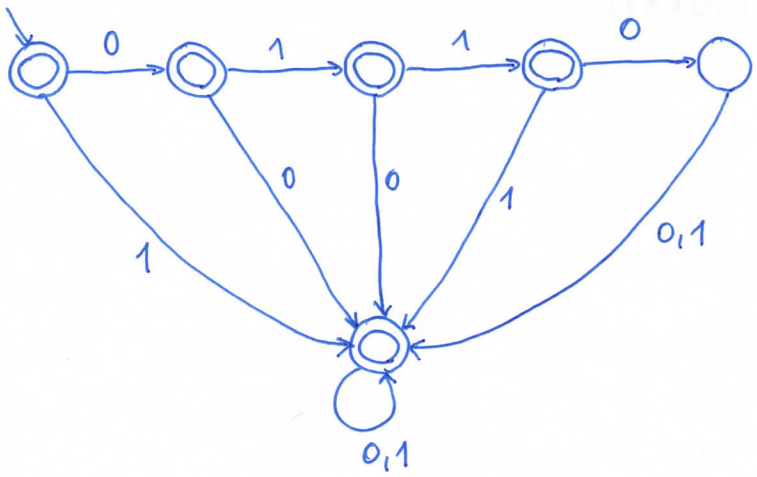
$((0+1)(0+1))^*$

4) Give a context-free grammar for the language L consisting of strings over $\{0,1\}$ where the number of 0's is congruent with 2 modulo 5. For example $001 \in L$, $1010 \in L$, but $0 \notin L$, $\epsilon \notin L$ and $10001 \notin L$. Explain the role of each non-terminal in your grammar.

T generates 1's
Z generates 0's

$S \rightarrow TZTZT$
$T \rightarrow 1T \mid \varepsilon$
$Z \rightarrow (TZ)^6T \mid 0$

5) We define over the alphabet $\Sigma = \{0,1\}$ the language L consisting of all possible strings except the string 0110. Describe a DFA for L.



6) We define over the alphabet $\Sigma = \{0,1\}$ the language L consisting of all possible strings except the string 0110. Write a regular expression for L.

The regular expression has three parts: strings of length at most 3, strings of length 4 different from 0110, and strings of length at least 5.

$(0+1+\varepsilon)^3 +$

$(\underbrace{0000+0001+\ldots+1111}_{15 \text{ strings}}) +$

$(0+1)^5 (0+1)^*.$

7) Give a context-free grammar for the language $L = \{a^i b^j c^k \mid i + k < j\}$. Explain the roles of each non-terminal in your grammar.

$a^i b^j c^k = \underbrace{a^i b^i}_{S_1} \underbrace{b^{j-i-k}}_{S_2} \underbrace{b^k c^k}_{S_3}$

$S \to S_1 S_2 S_3$

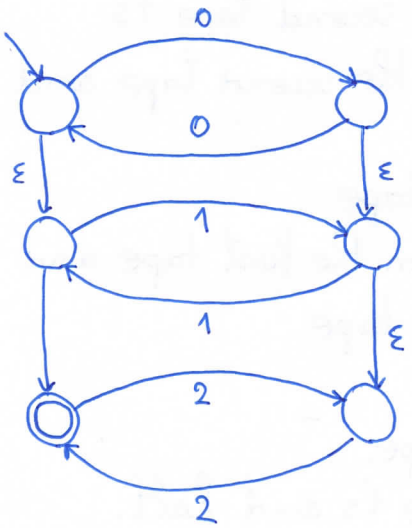
$S_1 \to a S_1 b \mid \varepsilon$

$S_2 \to b S_2 \mid \varepsilon$

$S_3 \to b S_3 c \mid \varepsilon$

8) Give an NFA for the language $L = \{0^n 1^m 2^p \mid p \equiv m + n \pmod 2\}$. For example $\epsilon \in L, 01 \in L, 22 \in L, 001112 \in L$, but $0 \notin L$ and $012 \notin L$.

We read a number of 0's, then a number of 1's, and then some 2's. We only need to keep track of the parities.



Problem 3 Turing machine construction (8 points)

Describe a two-tape Turing machine that computes the function $\lceil \log_2 n \rceil$. Given on the first tape as input the string 1^n (for any positive integer n), your machine should output the string $1^{\lceil \log_2 n \rceil}$ on its second tape. For example, given the input string 1111111111111 (thirteen 1's), your machine should output the string 1111 because $2^3 < 13 \leq 2^4$. Test your machine on the inputs 1111 and 11111.

Your description should be at the level of the description in the class for the Turing machine that accepts the set $\{1^n \mid n \text{ is a power of } 2\}$. In particular, do not give a list of transitions and do not draw state diagrams.

The idea is to simulate division by 2 by crossing out in each iteration every other 1 on the first tape. We need to treat separately the corner case when n is a power of 2. To detect this, for each pass through the first tape, we add a parity bit to the second tape. Each time we cross a 1 on the first tape, we flip the parity bit on the second tape.

repeat
{ 1. if there are only two 1's on the first tape
    2. if all bits on the second tape are 0's // n is a power of 2
        3a. make all bits on the second tape 1's
        3b. add a 1 on the second tape and halt.
    else // n is not a power of 2
        4a. make all bits on the second tape 1's
        4b. add two more 1's on the second tape and halt.
else
    5a add a 0 on the second tape
    5b cross out every other 1 on the first tape and flip the parity bit on the second tape.
}
until there are no 1's on the first tape.
6. make all the bits on the second tape 1's and halt.

Problem 4 NFA design (6 points).

Let $L \subseteq \Sigma^*$ be an arbitrary regular language. Prove that the language

$$\text{STUBS}(L) = \{x |\ xy \in L \text{ for some string } y \in \Sigma^*\}$$

is also regular. If you construct one machine $M' = (Q', \Sigma, \delta', q_0', F')$ from another machine $M = (Q, \Sigma, \delta, q_0, F)$, you must give precise definitions for Q, δ', q_0', and F'.

Stubs(L) contains all the strings in L and all the prefixes of the strings in L.

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$ for L, we construct an NFA $M' = (Q', \Sigma, \delta', q_0', F')$ for stubs(L). Let $Q_1 \subseteq Q$ be the set of all the states which are on paths from q_0 to any final state. We can find the states in Q_1 by a depth-first search.

We add a new final state q_{new} and add an ε-transition from any state in Q_1 to q_{new}. Thus

$Q' = Q \cup \{q_{new}\}$,

$q_0' = q_0$,

$F' = F \cup \{q_{new}\}$

Problem 5 Reduction (6 points). Prove that the language

$$\text{INCLUDE} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are Turing machines and } L(M_1) \subseteq L(M_2)\}.$$

is undecidable.

Assume we can decide include. We show that this implies we can decide Halt. Given any input $\langle M, w\rangle$ to the Halt problem, we construct an input $\langle M_1', M_2'\rangle$ for include as follows.

M_1': the machine which accepts any string

M_2': on any input y

1. run M on w.
2. if M halts, accept.

If $\langle M, w\rangle \in$ Halt, then $L(M_2') = \Sigma^*$, so $L(M_1') \subseteq L(M_2')$, which means that $\langle M_1', M_2'\rangle \in$ Include.

Conversely, if $\langle M, w\rangle \notin$ Halt (so M does not halt on w), then $L(M_2') = \emptyset$, and $L(M_1') \not\subseteq L(M_2')$. This implies that $\langle M_1', M_2'\rangle \notin$ include.

We conclude that the mapping $\langle M, w\rangle \to \langle M_1', M_2'\rangle$ is a reduction from Halt to include.

Problem 6 Enumeration and reduction (4 bonus points: 2 + 2). Consider the language

$L = \{\langle M \rangle \mid M \text{ is a Turing machine such that, for all strings } x, M \text{ on input } x \text{ halts within } |x|^2 \text{ steps } \}.$

Here $|x|$ denotes the length of a string x. We prove that L is not recognizable, but $\overline{L}$ is recognizable.

a) Design a recognizer for $\overline{L}$.

b) Prove that $\overline{\text{HALT}} \leq_m L$.

(Full bonus points to complete solutions. No points to partial solutions.)

a) $\overline{L} = \{\langle M \rangle \mid$ there is a string x such that M does not halt within $|x|^2$ steps on $x\}$.

Let $s_1, s_2, s_3, \ldots$ be a list of all possible strings in $\{0,1\}^*$. A recognizer for $\overline{L}$, on input $\langle M \rangle$, runs M for $|s_i|^2$ steps on input s_i, for $i = 1, 2, 3, \ldots$ If M does not halt when running for $|s_i|^2$ steps on s_i (for some i), then we accept.

If $M \in \overline{L}$, the recognizer we described will eventually be able to find a string x such that M does not halt within $|x|^2$ steps on x, at which point it will accept.

b) Let $\langle M, w \rangle$ be any input to $\overline{\text{Halt}}$. We construct a Turing machine M' which, on any input y, acts as follows:

1. simulate M on w for $|y|^2$ steps.
2. if M does not halt on w, M' halts.
   Otherwise hang.

If $\langle M, w \rangle \in \overline{\text{Halt}}$, then M does not halt on w, so M' halts on any input y in $|y|^2$ steps, which implies that $M' \in L$.

If $\langle M, w \rangle \notin \overline{\text{Halt}}$, so M halts on w, then M' hangs on any input y, which implies that $M' \notin L$.

We have reduced $\overline{\text{Halt}}$ to L.

Problem 7 NFA design (3 bonus points).
This problem is intended to be a challenge.

Let $L \subseteq \Sigma^*$ be an arbitrary regular language. Prove that the language

$$\text{REFLECT}(L) = \{w \in \Sigma^* \mid ww^R \in L\}$$

is also regular. If you construct one machine $M' = (Q', \Sigma, \delta', q_0', F')$ from another machine $M = (Q, \Sigma, \delta, q_0, F)$, you must give precise definitions for Q, δ', q_0', and F'. (Full bonus points to complete solutions. No points to partial solutions.)

Let M be a DFA for L. Define M' which simultaneously simulates transitions forward from q_0 and backward from any final state. If we arrive at the same state, we accept.

Formally

$Q' = \{q_{new}\} \cup \{(q_1, q_2) \mid q_1, q_2 \in Q\}$

$q_0' = q_{new}$

$\delta'(q_{new}, \varepsilon) = \{(q_{new}, q_f) \mid q_f \in F\}$

$\delta'((p, q), a) = \{(\delta(p, a), q') \mid \delta(q', a) = q\}$

$F' = \{(q, q) \mid q \in Q\}$