CS 3350 Automata, Computability, and Formal Languages
Spring 2020, Test 1

Name: _______

General comments:

- you are allowed up to 5 pages of handwritten notes;
- if you need extra pages, place your name on each extra page;
- the main goal of most questions is to show that you know the corresponding algorithms; so, if you are running of time, just follow the few first steps of the corresponding algorithm;
- each question will be graded on its own merit; so, for example, if when answering to the first part of the question, you got a wrong automaton, but on the second part, you correctly traced the new automaton, you will get full credit for the second part.

Good luck!

30/30

1-3. Let us consider an automaton for recognizing words starting with a pound sign #. For simplicity, let us only consider letters m and n. This automaton has 3 states: start (s), correct (c), and sink (si). Start is the starting state, correct is the only final state. The transitions are as follows:

- from the state s, symbol # leads to c, symbols m and n lead to si;
- from c, any symbol leads to c; and
- from si, every symbol leads to si.

10/10

1. Trace, step-by-step, how this finite automaton will check whether the following two strings start with #:

- the word #mn (which this automaton should accept) and
- the word m# (which this automaton should reject).

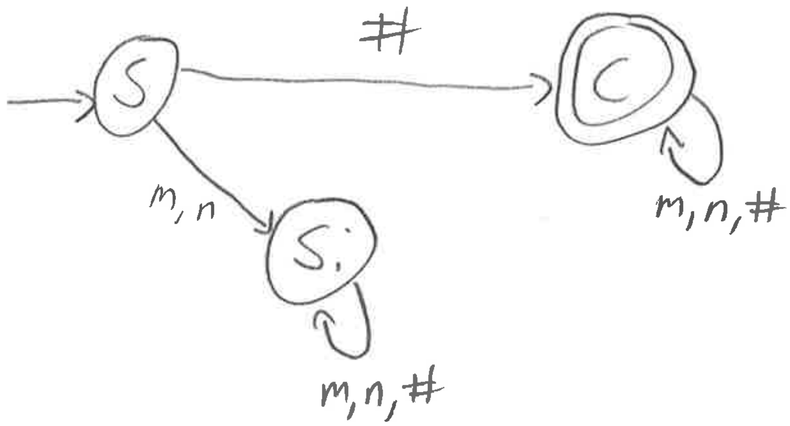
10/10

2. Use the above tracing to find the parts x, y, and z of the word #m#n corresponding to the Pumping Lemma. Check that the "pumped" word xyyz will also be accepted by this automaton.

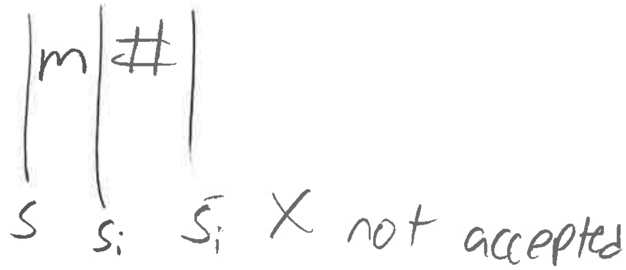
3. Write down the tuple <Q, Σ, δ, q_0, F> corresponding to this automaton:

10/10

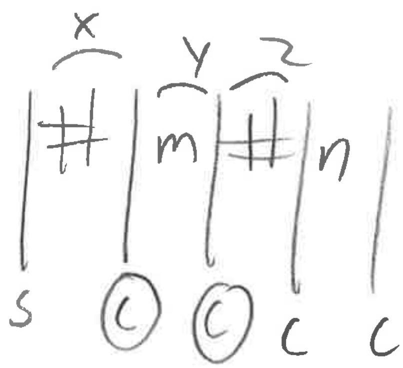
- Q is the set of all the states,
- Σ is the alphabet, i.e., the set of all the symbols that this automaton can encounter;
- δ: Q x Σ → Q is the function that describes, for each state q and for each symbol s, the state δ(q, s) to which the automaton that was originally in the state q moves when it sees the symbol s (you do not need to describe all possible transitions this way, just describe two of them);
- q_0 is the staring state, and
- F is the set of all final states.



1.



2.



Pumped word

3. $\langle Q, \Sigma, \delta, q_0, F \rangle$

$Q = \{ s, c, si \}$

$\Sigma = \{ \#, m, n \}$

$\delta =$

	s	c	si
#	c	c	si
m	si	c	si
n	si	c	si

$q_0 = s$

$F = \{ c \}$

4-5. Let A be the automaton described in Problem 1-3. Let B be an automaton that accepts all the strings that contain only # and m but not any other symbols. This automaton has two states: the start state which is also a final state, and the sink state. The transitions are as follows:

- from the start state, # or m leads back to the start state, n leads to the sink;
- from the sink state, any symbol leads back to the sink.

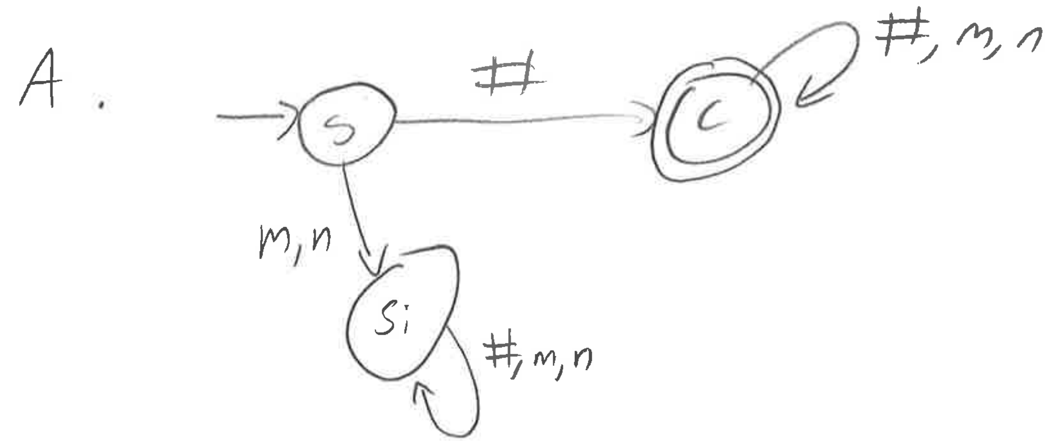
10/10 4. Use the algorithm that we had in class to describe the following two new automata:

- the automaton that recognizes the union A U B of the two corresponding languages, and
- the automaton that recognizes the intersection of the languages A and B.

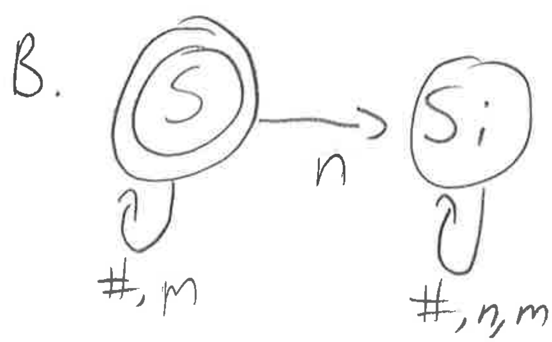
10/10 5. Test these two new automata step-by-step on the following words:

- test the union automaton on the example of the word m# (that it should accept);
- test the intersection automaton on the example of the words #mn (that it should reject).

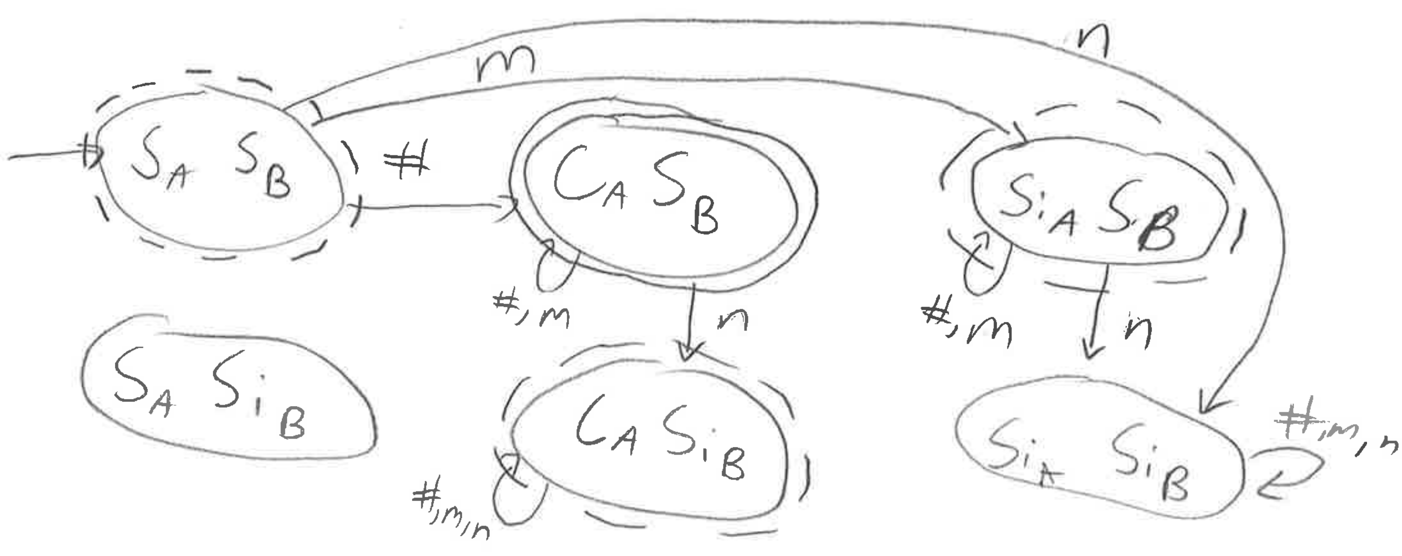
A.



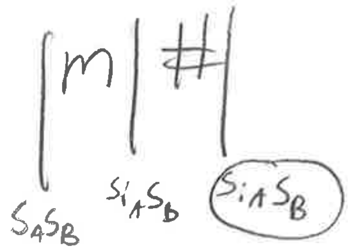
B.



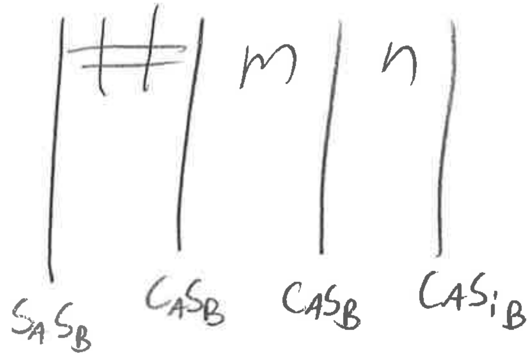
4.



5.



Accepted by Union



rejected by intersection

6-7.

8/10

6. Use the general algorithm that we learned in class to design a non-deterministic finite automaton that recognizes the language #(m U n U #)*:

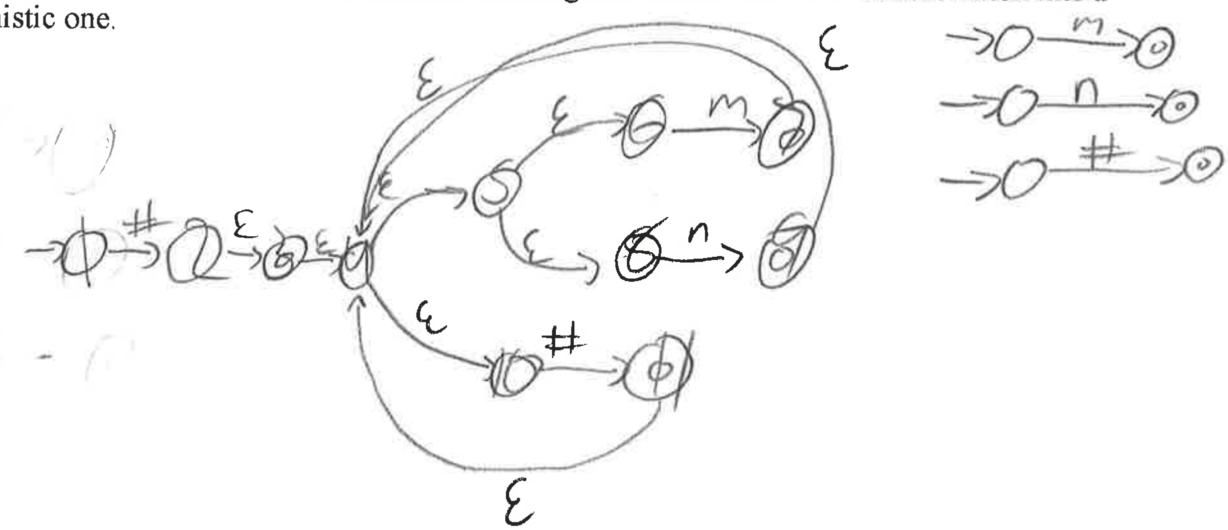
Needs step by step

- first, describe the automata for recognizing #, m, and n;
- then, combine them into the automata for recognizing the union m U n, (m U n) U #, and the Kleene star (m U n U #)*;
- finally, combine the automaton for # with an automaton for the Kleene star into an automaton for recognizing the desired composition of the two languages.

10/10

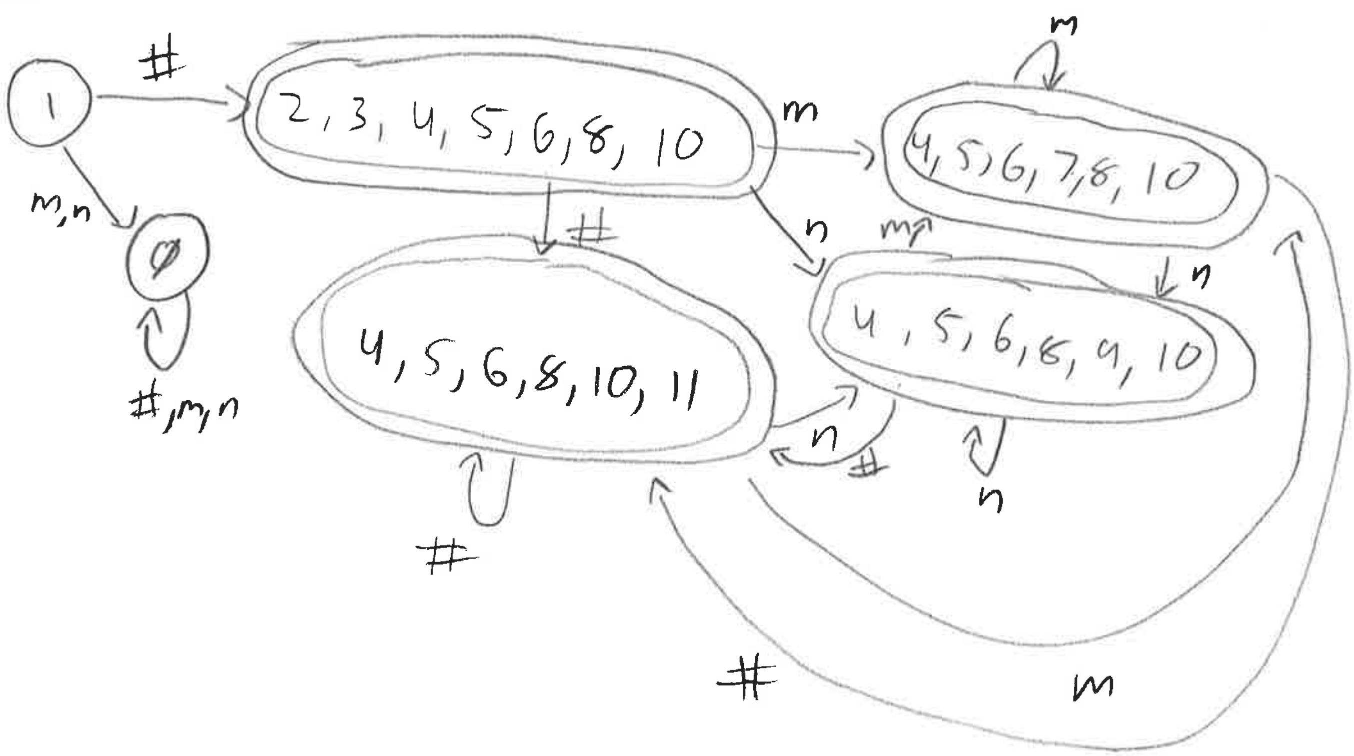
7. Use the general algorithm to transform the resulting non-deterministic finite automaton into a deterministic one.

6.



Deterministic

7.



6-7.

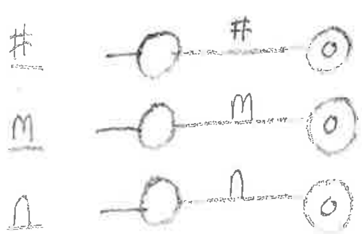
6. Use the general algorithm that we learned in class to design a non-deterministic finite automaton that recognizes the language #(m U n U #)*:

- first, describe the automata for recognizing #, m, and n;
- then, combine them into the automata for recognizing the union m U n, (m U n) U #, and the Kleene star (m U n U #)*;
- finally, combine the automaton for # with an automaton for the Kleene star into an automaton for recognizing the desired composition of the two languages.

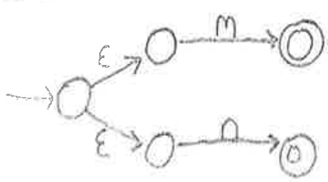
10/10

7. Use the general algorithm to transform the resulting non-deterministic finite automaton into a deterministic one.

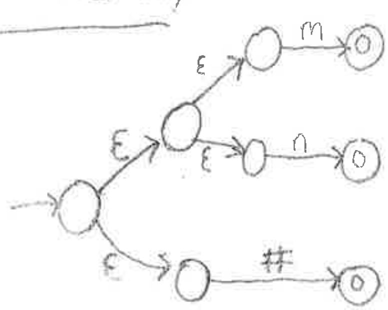
#(m U n U #)*



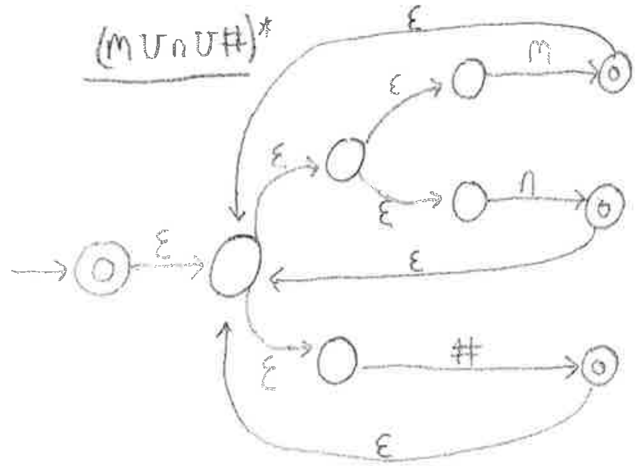
m U n



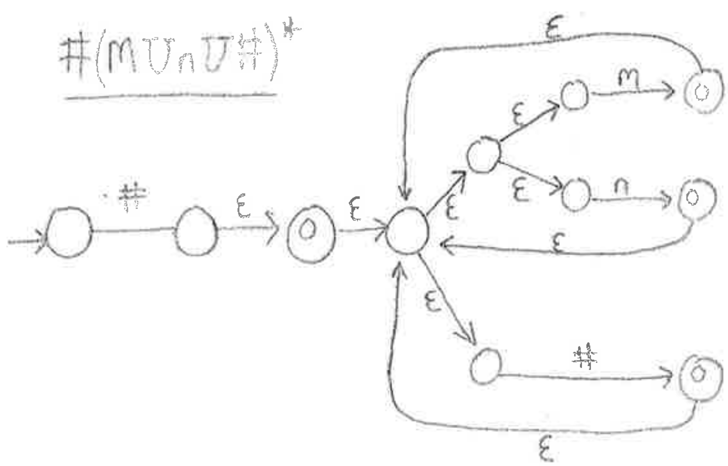
(m U n U #)



(m U n U #)*



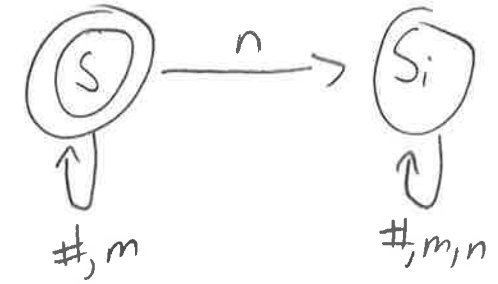
#(m U n U #)*

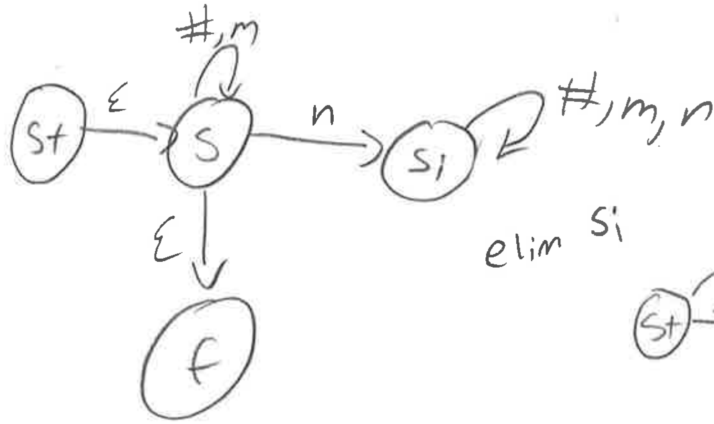


20/20

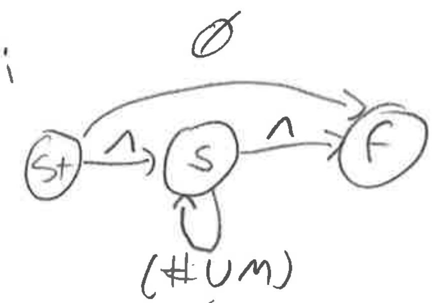
8-9. Use a general algorithm to transform the finite automaton B from Problem 4-5 into the corresponding regular expression.

8-9. B.





elim S_i



St → S
St → f
S → S
S → f

$$R'_{ij} = R_{ij} \cup (R_{ik} R_{kk}^* R_{kj})$$

$$R'_{St\,s} = R_{St\,s} \cup (R_{St\,S_i} R^*_{S_iS_i} R_{S_i s})$$

$$\Lambda \cup (\emptyset \ldots)$$

$$\Lambda$$

$$R'_{St\,f} = R_{St\,f} \cup (R_{St\,S_i} R^*_{S_iS_i} R_{S_i f})$$

$$\emptyset \cup (\emptyset \ldots)$$

$$\emptyset$$

$$R'_{SS} = R_{SS} \cup (R_{SS_i} R^*_{S_iS_i} R_{S_iS})$$

$$(\# \cup m) \cup (n(\# \cup n \cup m)^* \emptyset)$$

$$(\# \cup m) \cup \emptyset$$

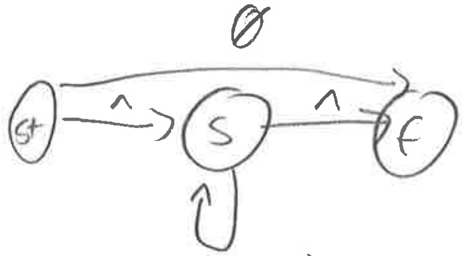
$$(\# \cup m)$$

$$R_{Sf} = R_{Sf} \cup (R_{SS_i} R^*_{S_iS_i} R_{S_if})$$

$$= \Lambda \cup (n(\# \cup n \cup m)^* \emptyset)$$

$$\Lambda \cup \emptyset$$

$$\Lambda$$



elim s

$$R'_{stf} = R_{stf} \cup (R_{sts} R_{ss}^{*} R_{sf})$$

$$\emptyset \cup (\lambda (\# \cup m)^{*} \lambda)$$

$$\emptyset \cup (\# \cup m)^{*}$$

$$(\# \cup m)^{*}$$

10/10

10. Prove that the language L of all the words that have equal number of m's and n's is not regular.

$L = \{$ eaual_num of m's and n's $\}$

Pumping lemma

$\exists p \forall w (len(w) \geq p \rightarrow x, y, z (w = xyz$ &

$len(y) > 0$ & $len(xy \leq p$ & $\forall i (xy^iz \in L))$

Assume L is regular

lets take $w = m^p n^p = \underbrace{m \ldots m}_{p} \underbrace{n \ldots n}_{p}$

$len(w) = 2p \geq p$

So, we can represent w as xyz. Since $len(xy) \leq p$, y is among the first p symbols, so y contains only m's. So, when we go from xyz to xyyz, we have more m's than n's. So xyyz is not in the language L. But by pumping lemma, xyyz is in the language L. Therefore, there is a contradiction so the language is not regular.