NAME: _____

SPIRE ID: _____

COMPSCI 250
Introduction to Computation
SOLUTIONS to Final Exam Fall 2025

D. A. M. Barrington and M. Golin 11 December 2025

DIRECTIONS:

- Answer the problems on the exam pages.
- There are **five** problems on pages **2-16**, some with multiple parts, for 120 points plus 10 extra credit. Final scale will be determined after the exam.
- Page **17** contains useful definitions and is given to you separately – do not put answers on it!
- If you need extra space use the back of a page – both sides are scanned.
- But, if you do write a solution on the back, you must **explicitly** add a note on the front stating that you used the back page. Otherwise, we might not see your solution on Gradescope.
- No books, notes, calculators, or collaboration.
- In case of a numerical answer, an arithmetic expression like "$2^{17} - 4$" need not be reduced to a single integer.
- Your answers must be LEGIBLE, and not cramped. Write only short paragraphs with space between paragraphs.

1	**/35**
2	**/10**
3	**/15+5**
4	**/40+5**
5	**/20**
Total	**/120+10**

Family Name: _____

Question 1 (Dog Proof, 35 total points) At Thanksgiving dinner, there were nine members of Dave's extended family. Dave and his wife Jessica were joined by his daughter Julia, son-in-law Will, grandson Graham, granddaughter Genevieve, his own dogs Blaze and Rhonda, and his daughter's dog Gwen.

Let $F = \{B, D, Ge, Gr, Gw, Je, Ju, R, W\}$ be the family members, and let $D = \{B, Gw, R\}$ be a subset of F containing the dogs. The menu included Dog Food, Green Bean Casserole, Pie, and Turkey — we will consider the set $M = \{DF, GBC, P, T\}$ of those items. Let $A \subseteq F \times M$ be a binary predicate so that $A(x, y)$ means "family member x ate menu item y".

We have six statements about the relation A:

(a, 10) Translations: Translate each statement as indicated.

- **Statement I:** (2, to symbols) Exactly one family member ate all four menu items.

  Solution: $\exists x : \forall y : (\forall z : A(y, z) \leftrightarrow (x = y))$

- **Statement II:** (1, to English) $A(Gr, GBC) \rightarrow (A(R, GBC) \wedge A(W, GBC))$.

  Solution: If Graham ate Green Bean Casserole, then so did both Rhonda and Will.

- **Statement III:** (1, to symbols) If either Will did not eat GBC or Rhonda ate GBC, then Graham ate GBC and Rhonda did not eat GBC.

  Solution: $(\neg(A(W, GBC) \vee A(R, GBC)) \rightarrow (A(Gr, GBC) \wedge \neg A(R, GBC))$

- **Statement IV:** (2, to English) $\forall x : \forall y : \forall z : [(x \in D) \wedge (y \in D)] \rightarrow (A(x, z) \leftrightarrow A(y, z))$.
  Solution: Any two dogs ate exactly the same menu items.

- **Statement V:** (2, to symbols) Dave, Julia, and Will each ate exactly the same menu items.

  Solution: $\forall m : (A(D, m) \leftrightarrow A(Ju, m)) \wedge (A(D, m) \leftrightarrow A(W, m))$

- **Statement VI:** (2, to English) $\forall x : A(x, T) \leftrightarrow (x \neq Je)$.

  Solution: Every family member ate Turkey if and only if they were not Jessica.

Grading notes: *The mean score was 8.1/10, but only 8% got 10/10, and 87% lost a point on Statement I. (50% missed SV, and 29% missed SIV.) If you write SI with an* $\leftrightarrow$*, it's important that the* $\forall z :$ *for the meal is* inside *the* $\leftrightarrow$*. A whole lot of you simply got the main quantifiers reversed, both in SI and SV.*

(b, 10) Boolean Proof:

Using Statement II and III *only*, prove that of the three boolean statements $p_1 = A(Gr, GBC)$, $p_2 = A(R, GBC)$, and $p_3 = A(W, GBC)$, p_1 and p_2 **are false** and that p_3 **is true**.

You may use either a truth table or a propositional proof. Please, for our ease in grading, use p_1, p_2, and p_3 rather than the original statements – you should write down your translation into this format before you start working.

Remember that you must prove *both* that your solution satisfies Statements II and III, *and* that no other solution satisfies both of them. Both follow from a complete truth table, but in a propositional proof you must make sure that both are included in your argument.

Solutions: These two statements translate to "$p_1 \to (p_2 \wedge p_3)$" and "$(\neg p_3 \vee p_2) \to (p_1 \wedge \neg p_2)$".

If we assume p_1, then p_2 and p_3 are true by Statement II. Since $\neg p_3 \vee p_2$ is then true, by Modus Ponens on Statement III it follows that $p_1 \wedge \neg p_3$, but this gives us both p_2 and $\neg p_2$, a contradiction. We may thus conclude that p_1 is false.

Since $p_1 \wedge \neg p_2$ is also false, by Modus Tollens on Statement III, $\neg p_3 \vee p_2$ must be false. By DeMorgan, then, we have $p_3 \wedge \neg p_2$.

Here is a truth table:

p_1	p_2	p_3	p_1	$\to$	$(p_2$	$\wedge$	$p_3)$	$(\neg p_3$	$\vee$	$p_2)$	$\to$	$(p_1$	$\wedge$	$\neg p_2)$	$SII \wedge SIII$
0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0
0	0	1	0	1	0	0	1	0	0	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1	1	1	0	0	0	0	0
0	1	1	0	1	1	1	1	0	1	1	0	0	0	0	0
1	0	0	1	0	0	0	0	1	1	0	1	1	1	1	0
1	0	1	1	0	0	0	1	0	0	0	1	1	1	1	0
1	1	0	1	0	1	0	0	1	1	1	0	1	0	0	0
1	1	1	1	1	1	1	1	0	1	1	0	1	0	0	0

Grading notes: *The mean score was 8.91/10, and 63% of you got full marks for truth tables. Another 6% got full marks for a propositional proof, and another 7% got 9/10 for an otherwise propositional proof but didn't check that their solution satisfied the statements. The majority of the truth tables had their rows were in the standard order, or the reverse of the standard order, but an annoying number of you had them in a different order. As usual, there were a number of different valid ways to carry out the propositional proof.*

Family Name: _____

(c, 15) Predicate Proof:

Using any or all of Statements I, II, III, IV, V, and VI, **prove that Genevieve ate Dog Food**. We are providing a table for all 36 possible truth values of the relation A, but you are *not* asked to fill out this table completely, and we *do not* guarantee that all 36 values are determined from the six statements. We have already filled in the three facts you know from part (B), where **0** means false, and **1** means true.

$x \backslash y$	DF	GBC	P	T
B				
D				
Ge				
Gr		**0**		
Gw				
Je				
Ju				
R		**0**		
W		**1**		

If your answer involves the use of a quantifier proof rule, and many of them should, make clear which rule you are using and when. Using complete sentences will generally make your answers more readable. If your later statements follow from previous statements you have proved, make clear which is the premise and which the conclusion. Quantifier rules may be expressed in either symbols or English. If you are using your own translation of one of the premises in your proof, check again that your statement is equivalent to the original.

Solution:

By Instantiation on Statement I, let X be the family member who ate all four items. We will prove that X is Genevieve, from which we can use Specification on the menu item DF to prove that she ate Dog Food.

X cannot be Graham, since II and III imply that he did not eat

X cannot be Jessica, since she did not eat Turkey, while X did.

If X were any of Dave, Julia, or Will, Statement IV would imply that one of those other two family items would also have eaten all four menu items, contradicting Statement I which says that only one family member did so.

If X were any of the dogs, we know by Statements II and III that Rhonda did not eat GBC, and by Statement IV neither of the other dogs did so, so since X ate GBC, none of them are X. We can also make a similar argument to the one above, that if one of the dogs ate all four menu items, then they all did, contradicting Statement I.

This eliminates eight of the nine candidates to be X, so X is Genevieve and she ate the Dog Food as well as the other three menu items.

(Note that Statements I-V do not tell us who else ate Dog Food, or who else ate Pie, or whether Jessica ate GBC. (We do know that Dave, Julia, and Will either all ate Pie or all ate Dog Food, but not both.)

Here's a table of the facts given by the six statements. The question marks could represent any boolean values, and the booleans a, b, c, and d could have any values, except that b and d cannot both be true.

$x\backslash y$	DF	GBC	P	T
B	a	0	c	1
D	b	1	d	1
Ge	1	1	1	1
Gr	?	**0**	?	1
Gw	a	0	c	1
Je	?	?	?	0
Ju	b	1	d	1
R	a	**0**	c	1
W	b	**1**	d	1

The actual foods eaten by my extended family that day were:

$x\backslash y$	DF	GBC	P	T
B	1	0	0	1
D	0	1	1	1
Ge	1	1	1	1
Gr	0	**0**	1	1
Gw	1	0	0	1
Je	0	1	1	0
Ju	0	1	1	1
R	1	**0**	0	1
W	0	**1**	1	1

Grading notes: *The mean score was 10.86/15, with 44% of you getting full marks and an additional 15% getting 12/15. I gave 6/15 in general if you didn't get to the point that the only way to prove that anyone ate any dog food was to identify the person specified in Statement I. In order to eliminate Graham, I needed some statement in your text to assert that it wasn't him. "Everyone else has some food they are known not to have eaten" was fine, but just saying that "everyone else has been eliminated" wasn't if you were just relying on the 0 in the table.*

Family Name: _____

Question 2 (10 points): (Induction 2) For naturals n, let A_n be defined by

$$A_0 = 2, \quad A_1 = 6 \quad \text{and for all } n \geq 1, \;\; A_{n+1} = 6A_n - 8A_{n-1}.$$

Examples: $A_2 = 6A_1 - 8A_0 = 36 - 16 = 20$ and $A_3 = 6A_2 - 8A_1 = 6 \cdot 20 - 8 \cdot 6 = 72$.

Prove *by induction*, that for *all* naturals n, $A_n = 4^n + 2^n$.

As a validity check. note that $A_2 = 20 = 4^2 + 2^2$ and $A_3 = 72 = 64 + 8 = 4^3 + 2^3$.

a) First write your induction hypothesis in the box below. This should be in the form $P(x)$, where you *must* explicitly explain what x is and write an unambiguous statement of $P(x)$.

Solution:
$P(n) : A_n = 4^n + 2^n$ for all naturals n.

(b) Next, write your base case(s) in the box below.

Solution:
Base Cases: $n = 0$ and $n = 1$.
This is correct because $A_0 = 2 = 1 + 1 = 4^0 + 2^0$ and $A_1 = 6 = 4 + 2 = 4^1 + 2^1$.

(c) Finally, provide your induction step. This step will be marked on how clear and mathematically precise your proof is. Ambiguous explanations or explanations missing details will have points deducted.

We use strong induction.

Assume that, for $n \geq 1$, $P(i)$ is true for all $i \leq n$.
We need to prove that this implies $P(n+1)$:

$$\begin{aligned} A_{n+1} &= 6A_n - 8A_{n-1} && \textbf{(Definition)} \\ &= 6\left(4^n + 2^n\right) - 8\left(4^{n-1} + 2^{n-1}\right) && \textbf{(two uses of IH)} \\ &= 4^n(6-2) + 2^n(6-4) && \textbf{(algebraic manipulation)} \\ &= 4^{n+1} + 2^{n+1}, \end{aligned}$$

i.e., that $A_{n+1} = 4^{n+1} + 2^{n+1}$, i.e., $P(n+1)$.

Grading notes: *The mean score was 8/10 with a median of 9.5. 43% received full marks and 73% received at least an 8/10 The most common error was not getting the base cases correct (see marking note below for more details).*

Marking Note 2(i): Most students got this correct. The only small error that occasionally occurred was in identifying the base cases. Because A_{n+1} required knowing BOTH A_n and A_{n-1}, this needed two base cases. Otherwise, it would be impossible to calculate, e.g., A_2. Base cases of $n = 1, 2$, would not be possible because then it would not be possible to use the induction step to prove $P(2)$.

Family Name: _____

Question 3 (a,5): (Induction 1)

Note: The original printed version of the exam contained some typos in the expansion of NDL and in the definitions of A_n and B_n. These were corrected in real-time during the exam and have been corrected in this version.

We will say that a word w contains a *doubled letter* if some character in the word appears twice in a row. For example $w_1 = abccababa$ contains the doubled letter cc, while $w_2 = abcabcba$ does not contain any doubled letter.

For a fixed alphabet Σ let language NDL (no doubled-letter) be the set of all words in Σ^* that do not contain a double letter. For our problem, fix $\Sigma = \{a, b\}$, then

$$NDL = \{\lambda, a, b, ab, ba, aba, bab, abab, baba, ababa, babab, \ldots\}.$$

Furthermore, for all positive naturals n, define

$$A_n = \{w : \ w \in NDL, |w| = n \text{ and } w \text{ ends in } a\} \quad \text{and} \quad B_n = \{w : \ w \in NDL, |w| = n \text{ and } w \text{ ends in } b\}.$$

As examples, $A_2 = \{ba\}$, $B_2 = \{ab\}$, $A_3 = \{aba\}$, $B_3 = \{bab\}$.
Recall that if X is a set, $|X|$ is the number of items in the set.

Let $P(n)$ be the statement "$|A_n| = |B_n| = 1$", defined for all positive naturals n.

Prove by induction that, for all positive naturals n, $P(n)$ is true.

Note that, for this problem, we have already given you the induction hypothesis, $P(n)$.

(i) Now, write your base case(s) in the box below.

Solution: The base case is $n = 1$. $A_1 = \{a\}$ and $B_1 = \{b\}$ so $|A_1| = |B_1| = 1$.

(ii) Finally, provide your induction step, i.e. proving that assuming $P(n)$ implies $P(n + 1)$.

This step will be marked on how clear and mathematically precise your proof is. Ambiguous explanations will have points deducted.
Be sure to clearly describe your induction goal. If you run out of space, continue the proof on the back page (with a note stating that you are writing on the back).

Solution:

There are two different solution approaches.

The first constructs elements in, A_{n+1}, by adding an a to the end of every string in B_n, and elements in, B_{n+1}, by adding a b to the end of every string in A_n.

The second constructs elements in, A_{n+1} and B_{n+1} by adding appropriate elements, respectively, to the start of every string in A_n and B_n.

Solution Method 1: Showing $A_{n+1} = B_n a$ and $B_{n+1} = A_n b$.

Note that if $w \in A_{n+1}$ for $n \geq 1$, then $w = va$ for some v and, in particular

$$\begin{array}{lll} va \in A_{n+1} & \text{iff} & |va| = n+1 \text{ and } va \in NDL \\ & \text{iff} & |v| = n, \text{ and } v \in NDL \text{ and } v \text{ ends in a } b \\ & \text{iff} & v \in B_n. \end{array}$$

Using regular expression notation, we have just shown $A_{n+1} = B_n a$. So there is a one-to-one correspondence between the words in A_{n+1} and the words in B_n, so $|A_{n+1}| = |B_n|$.

Example. Set $v = bab$ and $w = va = baba$, Then $A_4 = \{w\}$ and $B_3 = \{v\}$ so $A_4 = \{w\} = \{v\}a = B_3 a$.

Similarly, if $w \in B_{n+1}$ for $n \geq 1$, then $w = vb$ for some v and, in particular

$$\begin{array}{lll} vb \in B_{n+1} & \text{iff} & |vb| = n+1 \text{ and } vb \in NDL \\ & \text{iff} & |v| = n \text{ and } v \in NDL \text{ and } v \text{ ends in an } a \\ & \text{iff} & v \in A_n. \end{array}$$

Using regular expression notation, we have just shown $B_{n+1} = A_n b$. So, there is a one-to-one correspondence between the words in B_{n+1} and the words in A_n, so $|B_{n+1}| = |A_n|$.

Now, assume $n \geq 1$ and $P(n)$ is true, i.e., $|B_n| = 1 = |A_n|$.

Plugging the facts derived above into the IH we get

$$|A_{n+1}| = |B_n| = 1 = |A_n| = |B_{n+1}, |$$

and we have proven $P(n+1)$.

Note that the if and only if statements proved both directions. This was necessary. See Marking note 3(a)1 below.

Solution Method 2: A more complicated proof would be to add a character at the start. More explicitly, for $n \geq 1$, assume $P(n)$.

Then

$$w \in A_{n+1} \quad \text{iff} \quad w = xv,\ v \in A_n,\ \text{and}\ \ x \text{ is not the same as the first character in } v. \qquad (1)$$

Example. $A_4 = \{w = baba\}$. $A_3 = \{v = aba\}$. *The first character in* v *is* $'a'$ *so* $x =' b'$ *and* $w = bv$.

The reason that $v \in A_n$ on the RHS is that we must have $|v| = n$, $v \in NDL$ and v ends in a.

Since, by the IH, there is exactly one string $v \in A_n$, x is uniquely defined and there is exactly one string $w = xv \in A_{n+1}$. So $|A_{n+1}| = |A_n| = 1$

Similarly, for $n \geq 1$

$$w \in B_{n+1} \quad \text{iff} \quad w = xv,\ v \in B_n,\ \text{and}\ \ x \text{ is not the same as the first character in } v. \qquad (2)$$

The reason that $v \in B_n$ on the RHS is that we must have $|v| = n$, $v \in NDL$ and v ends in b.

Since, by the IH, there is exactly one string $v \in B_n$, x is uniquely defined and there is exactly one string $w = xv \in B_{n+1}$. So $|B_{n+1}| = |B_n| = 1$

Note that Equations (1) and (2) are if and only if statements in which we had to prove both directions. As in the proof using the first method, this was necessary. See Marking note 3(a)1 below.

Grading notes: *The mean score was 2.6/5 with a median of 2.5. Only 7% got full marks with an additional 26% getting 4/5 or 4.5/5. The most common error by far was only stating/proving one direction of the if and only if proof (see Marking note 3(a)1 below for more details).*

Marking Note 3(a) 1 The first method worked by showing a one-to-one correspondence between items in A_{n+1} and items in B_n (and between items in B_{n+1} and items in A_n). While not hard, many submitted "proofs" were incomplete. For example,

- they might have stated "every element in A_{n+1} must be generated by an element in B_n". One bug here is that this lacks (required) justification. The statement was also ambiguous in what it meant by "generated".
- they might have stated "every element in A_{n+1} must be generated by adding an "a" to the end of an element in B_n".
  While this might look less ambiguous it still suffers from the fact that it lacks justification. A-Priori, it might be possible to generate a string in A_{n+1} using a different method. You must PROVE that this statement is correct.
  In fact, the proof that this statement is correct is a MAIN part of the induction proof.

A second issue with the formulations above is that they are not "if and only if'. They stated

- "every element in A_{n+1} can be generated by an element in B_n followed by an a", which means $A_{n+1} \subseteq B_n a$, or they stated
- "every element in B_n can generated from an element in A_{n+1} by removing the last a from the string in A_{n+1},$''$ which means $B_n a \subseteq A_n a$.

Neither of these were fully sufficient. The proof needed a one to one correspondence between A_{n+1} and B_n, i.e., stating and proving the equivalent of $A_{n+1} = B_n a$.

Both directions needed to be explicitly acknowledged and justified. Otherwise this would not be a one-to-one correspondence.

The same issue of requiring a two-directional proof, i.e., an **iff** statement, was required to get the one-to-one correspondence when using the second method as well.

Marking Note 3(a) 2: Some solutions started off saying that they were using "structural induction". This was wrong, since the problem explicitly asked to prove $P(n) \rightarrow P(n+1)$, which is induction on the naturals!
Points were not deducted for this error, though, unless it was combined with some other error.

Marking Note 3(a) 3: some proofs said something like:
"Since the NDL condition requires that characters alternate between a and b, there is only one string of length n that ends (or starts) in an a (in a b)."

While this is perfectly correct, this would be a combinatorial proof and not an induction proof, and the question explicitly required an induction proof.

In fact, the combinatorial property (alternating) upon which the combinatorial proof relies must be proven to be correct before it can be used. The proof of its validity is (you guessed it) via induction!

Family Name: _____

Question 3 (b, 10): (Induction)

Note: The original printed version of the exam contained some typos in the definitions of A_n B_n and C_n. These were corrected in real-time during the exam and have been corrected in this version.

Now let $\Sigma = \{a, b, c\}$. Then

$NDL = \{\lambda, a, b, c, ba, ca, ab, cb, ac, bc, aba, cba, aca, bca, bab, cab, acb, bcb, bac, cac, abc, cbc, \ldots\}$.

Furthermore, for all positive naturals n, define

$A_n = \{w : \ w \in NDL, |w| = n \text{ and } w \text{ ends in } a\}, \quad B_n = \{w : \ w \in NDL, |w| = n \text{ and } w \text{ ends in } b\},$

$$C_n = \{w : \ w \in NDL, |w| = n \text{ and } w \text{ ends in } c\},$$

As examples, $A_2 = \{ca, ba\}$, $B_2 = \{ab, cb\}$, $C_2 = \{ac, bc\}$.

Prove by induction that, for all positive naturals n, $|A_n| = |B_n| = |C_n| = 2^{n-1}$.

(i) First write your induction hypothesis in the box below. This should be in the form $P(x)$, where you *must* explicitly explain what x is and write an unambiguous statement of $P(x)$.

Solution: $P(n) : |A_n| = |B_n| = |C_n| = 2^{n-1}$. n is a positive natural.

(ii) Next, write your base case(s) in the box below.

Solution: The base case is $n = 1$. $A_1 = \{a\}$, $B_1 = \{b\}$ and $C_1 = \{c\}$
so $|A_1| = |B_1| = |C_1| = 1 = 2^0 = 2^{1-1}$.

(iii) Finally, provide your induction step. This step will be marked on how clear and mathematically precise your proof is. Ambiguous explanations will have points deducted.
Be sure to clearly describe your induction goal. If you run out of space, continue the proof on the back page (with a note stating that you are writing on the back).

Solution:

Assume $P(n)$.

As in the previous problem, there are two different solution approaches. The first constructs elements in, e.g., A_{n+1}, by adding an a to the end of every string in $B_n + C_n$.

The second constructs e.g., A_{n+1}, by adding each of two possible characters to the start of each string in A_n. Here are the details

Solution 1: Showing $A_{n+1} = B_n a + C_n a$, $B_{n+1} = A_n b + C_n b$ and $C_{n+1} = A_n c + B_n c$

$$\begin{array}{lll} w \in A_{n+1} & \text{iff} & w = w'a \text{ where } w' \in NDL, |w'| = n \text{ and } w' \text{ doesn't end in an } a \\ & \text{iff} & w = w'a \text{ where } w' \in B_n + C_n \end{array}$$

Using regular expression notation, we have shown $A_{n+1} = B_n a + C_n a$.

Since, **by the induction hypothesis** $|B_n| = |C_n| = 2^{n-1}$ and by definition, $B_n a \cap C_n a = \emptyset$, we have $|A_{n+1}| = |B_n| + |C_n|$ which yields

$$|A_{n+1}| = |B_n| + |C_n| = 2^{n-1} + 2^{n-1} = 2^n = 2^{(n+1)-1}$$

The proof that $|B_{n+1}| = 2^{(n+1)-1}$ and $|C_{n+1}| = 2^{(n+1)-1}$ is almost exactly the same. It wasn't necessary to write them completely as long as you explained why they were almost the same.

For completeness, we WILL write out both cases.

$$\begin{array}{lll} w \in B_{n+1} & \text{iff} & w = w'b \text{ where } w' \in NDL, |w'| = n \text{ and } w' \text{ doesn't end in a } b \\ & \text{iff} & w = w'b \text{ where } w' \in A_n + C_n \end{array}$$

Using regular expression notation, we have shown $B_{n+1} = A_n b + C_n b$.

Since, **by the induction hypothesis** $|A_n| = |C_n| = 2^{n-1}$ and by definition, $A_n b \cap C_n b = \emptyset$, we have $|B_{n+1}| = |A_n| + |C_n|$ which, by the IH is

$$|B_{n+1}| = |A_n| + |C_n| = 2^{n-1} + 2^{n-1} = 2^n = 2^{(n+1)-1}$$

Finally,

$$\begin{array}{lll} w \in C_{n+1} & \text{iff} & w = w'c \text{ where } w' \in NDL, |w'| = n \text{ and } w' \text{ doesn't end in a } c \\ & \text{iff} & w = w'c \text{ where } w' \in A_n + B_n \end{array}$$

Using regular expression notation, we have shown $C_{n+1} = A_n b + B_n b$.

Since, **by the induction hypothesis** $|A_n| = |B_n| = 2^{n-1}$, and by definition, $A_n c \cap B_n c = \emptyset$, we have $|C_{n+1}| = |A_n| + |B_n|$ which, by the IH is

$$|C_{n+1}| = |A_n| + |B_n| = 2^{n-1} + 2^{n-1} = 2^n = 2^{(n+1)-1}$$

We have therefore proven that $P(n)$ implies $P(n+1)$, i.e.,

$$|A_{n+1}| = |B_{n+1}| = |C_{n+1}| = 2^{(n+1)-1}.$$

As an example of the construction, $B_2 = \{ab, cb\}$ and $C_2 = \{ac, bc\}$.

$$A_3 = \{aba, cba, aca, bca\} = \{aba, cba\} + \{aca, bca\} = B_2 a + C_2 a.$$

Solution 2:

$w \in A_{n+1}$ iff $w = xv$ where $v \in NDL$, $|v| = n$, v ends in a and x is not the first character in v
iff $w = xv$ where $v \in A_n$ and x is not the first character in v

We can therefore construct every string in A_{n+1} uniquely as follows:

1. Let $v \in A_n$ be arbitrary.
2. Let x_0 be the first character in v.
3. Let $x_1 \neq x_2$ be the two characters in $\{a, b, c\}$ different from x_0.
4. Define $v' = x_1 v$ and $v'' = x_2 v$.

From the iff statements above, above, v', v'' are both in A_{n+1} and every w in A_{n+1} can be built as either v' or v'' from some $v \in A_n$.

The IH tells us that $|A_n| = 2^{n-1}$, which immediately implies

$$|A_{n+1}| = 2|A_n| = 2 \cdot 2^{n-1} = 2^n = 2^{(n+1)-1}.$$

The proof that $|B_{n+1}| = 2^{(n+1)-1}$ and $|C_{n+1}| = 2^{(n+1)-1}$ is almost exactly the same. Just replace A_{n+1} and A_n with B_{n+1} and B_n (or C_{n+1} and C_n) and keep the rest of the proof the same to get $|B_{n+1}| = 2^{(n+1)-1}$ and $|C_{n+1}| = 2^{(n+1)-1}$.

As an example of the construction, consider $A_2 = \{ba, ca\} = \{v_1, v_2\}$.
v_1 *start with a* $'b'$ *and* v_2 *starts with a* $'c'$*, so*

$$A_3 = \{av_1, cv_1, av_2, bv_2\} = \{aba, cba, aca, bca\}.$$

Grading notes: *The mean score was 5/10 with a median of 4. Only 7% got full marks with an additional 25% getting between 8 and 9.5. Again, the most common error by far was only stating/proving one direction of the if and only if proof (see Marking note 3(b)1 below for more details).*

Marking Notes: The issues that arose here were almost the same as those that arose in the previous question. We only sketch them below. Please read the marking notes for the previous question for the full details.

Marking Note 3(b) 1: It was necessary to show one-to one correspondences between the appropriate sets. This required proving if and only if statements. Only showing one direction had some points deducted. This was explained in great detail in the associated rubrics. Also, justifications were required for every statement.

As an example, if using method 1, it was necessary to PROVE $A_{n+1} = B_n a + C_n a$. Just proving $A_{n+1} \subseteq B_n a + C_n a$ or $A_{n+1} \supseteq B_n a + C_n a$, was not suffcient.

Marking Note 3(b) 2: Some solutions started off saying that they were using "structural induction". This was wrong, since the problem explicitly asked to prove $P(n) \rightarrow P(n+1)$, which is induction on the naturals!
Points were not deducted for this error, though, unless it was combined with some other error.

Marking Note 3(b) 3: some proofs said something like (only stated for A_n. Argument was the same for B_n and C_n):

Consider a string $v \in A_n$. Let v_i, $i = 1, 2, \ldots, n$, be the character in v at location i. Then $v_n =$ 'a'. Given the value of v_i, any of the two other characters in $\{a, b, c\}$ not equal to v_i can be chosen for v_{i-1}. Since, by the NDL restriction, there are two possible choices for each of $v_{n-1}, v_{n-2}, \ldots, v_1$, there are 2^{n-1} possible ways of building v. So $|A_n| = 2^{n-1}$

This is a combinatorial proof and, while it would be a valid argument in a combinatorics class, our problem explicitly asked for an induction proof, so it was not acceptable here.

At a more core level,, the validation of the correctness of the combinatorial proof is via induction. Our class is at a lower proof level than those other classes and we are proving the correctness of the techniques that those higher level classes use.

Family Name: _____

Question 3 (c, 5): (XC) Note: The original printed version of the exam contained some typos in the definitions of A_n and B_n. These were corrected in real-time during the exam and have been corrected in this version.

We now say that a word w contains a *tripled letter* if some character in the word appears three in a row. For example $w_1 = abbaaabaa$ contains the tripled letter aaa, while $w_2 = abbaabbabbaa$ does not contain any tripled letter.

For a fixed alphabet Σ we let language NTL (no tripled-letter) be the set of all words in Σ^* that do not contain a triple letter.

Now set $\Sigma = \{a, b\}$. Then

$$NTL = \{\lambda, a, b, aa, ab, ba, bb, baa, aba, bba, abb, bab, aab \ldots\}.$$

Furthermore, for all positive naturals n, define

$$A_n = \{w : \ w \in NTL, |w| = n \text{ and } w \text{ ends in } a\} \quad \text{and} \quad B_n = \{w : \ w \in NTL, |w| = n \text{ and } w \text{ ends in } b\}.$$

As examples, $A_2 = \{aa, ba\}$, $B_2 = \{ab, bb\}$, $A_3 = \{baa, aba, bba\}$, $B_3 = \{abb, bab, aab\}$.

Recall the Fibonacci numbers F_n are defined on the naturals n by: $F_0 = 0$, $F_1 = 1$, and,

$$\forall n \geq 1, \ F_{n+1} = F_n + F_{n-1}. \tag{3}$$

The next four Fibonacci numbers are then $F_2 = 1$, $F_3 = 2$, $F_4 = 3$ and $F_5 = 5$.

Prove by induction that, for all positive naturals, $|A_n| = |B_n| = F_{n+1}$.

(i) First write your induction hypothesis in the box below. This should be in the form $P(x)$, where you *must* explicitly explain what x is and write an unambiguous statement of $P(x)$.

Solution: $P(n) : |A_n| = |B_n| = F_n$. n is a positive natural.

(ii) Next, write your base case(s) in the box below.

Solution: The base cases are $n = 1$ and $n = 2$
As shown above $|A_1| = |B_1| = 1$ and $|A_2| = |B_2| = 2$.
So $|A_1| = |B_1| = 1 = F_2$ and $|A_2| = |B_2| = 2 = F_3$

(iii) Finally, provide your induction step. This step will be marked on how clear and mathematically precise your proof is. Ambiguous explanations will have points deducted.
Be sure to clearly describe your induction goal. If you run out of space, continue the proof on the back page (with a note stating that you are writing on the back).

Solution: The proof will be by strong induction. Suppose $n \geq 2$.

- A word $w \in A_{n+1}$ if and only if (i) $w = w'a$ or (ii) $w = w'aa$, where $w' \in NTL$ and w doesn't end in an a.

- In (i) and (ii) $w' \in NDL$ and ends in a b which means that $w \in A_{n+1}$ if and only if (i) $w = w'a$ where $w' \in B_n$ or (ii) $w = w'aa$, where $w' \in B_{n-1}$.
- This implies $A_{n+1} = B_n a + B_{n-1}aa$ so

$$|A_{n+1}| = |B_n| + |B_{n-1}|. \tag{4}$$

- **By the induction hypothesis** (applied twice) this gives the required

$$|A_{n+1}| = |B_n| + |B_{n-1}| = F_{n+1} + F_n = F_{n+2} = F_{(n+1)+1}.$$

Similarly,

- A word $w \in B_{n+1}$ if and only if (i) $w = w'b$ or (ii) $w = w'bb$, where $w' \in NTL$ and w doesn't end in an b.
- In (i) and (ii) $w' \in NDL$ and ends in a a which means that $w \in B_{n+1}$ if and only if (i) $w = w'b$ where $w' \in A_n$ or (ii) $w = w'bb$, where $w' \in A_{n-1}$.
- This implies $B_{n+1} = A_n b + A_{n-1}bb$ so

$$|B_{n+1}| = |A_n| + |A_{n-1}|. \tag{5}$$

- **By the induction hypothesis** (applied twice) this gives the required

$$|B_{n+1}| = |A_n| + |A_{n-1}| = F_{n+1} + F_n = F_{n+2} = F_{(n+1)+1}.$$

Combining the two parts, we have therefore shown that, for $n \geq 2$, $P(n-1) \wedge P(n) \rightarrow P(n+1)$.

Example of the construction $B_1 = \{b\}$, $B_2 = \{ab, bb\}$,

$$A_3 = \{baa, aba, bba\} = \{baa\} + \{aba, bba\} = B_1 a + B_2 aa.$$

Grading notes: *The mean score was 1.5/5 with a median of 2 (with around 30% of the class not attempting to answer this XC).*

The most common errors were only stating/proving one direction of the if and only if proof and also not realizing that there had to be two base cases.

Marking Note 3(c) 1: The same issue occurred here as in parts (a) and (b). For both equations (4) and (5) it was necessary to justify both directions. For example, for A_{n+1}, it was not sufficient to show just that $A_{n+1} \subseteq B_n + B_{n-1}$ or that $A_{n+1} \supseteq B_n + B_{n-1}$ (and proofs using just words and not mathematical formalism tended to show only one direction).

Marking Note 3(c) 2: Another common issue was either not noticing there had to be two base cases or getting at least one of the two base cases incorrect.

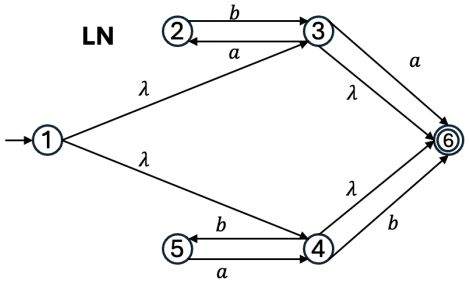
Family Name: _____

Question 4 (40+5 points total):

This question involves several of the constructions from Kleene's Theorem. We are actually using the language NDL as in Question 3(A). What this means is that you may have extra chances to confirm that your new objects have the same language as before.

We begin with an ordinary λ-NFA LN, with alphabet $\Sigma = \{a, b\}$, state set $\{1, 2, 3, 4, 5, 6\}$, start state 1, final state set $\{6\}$, and transition relation (with ten total transitions):

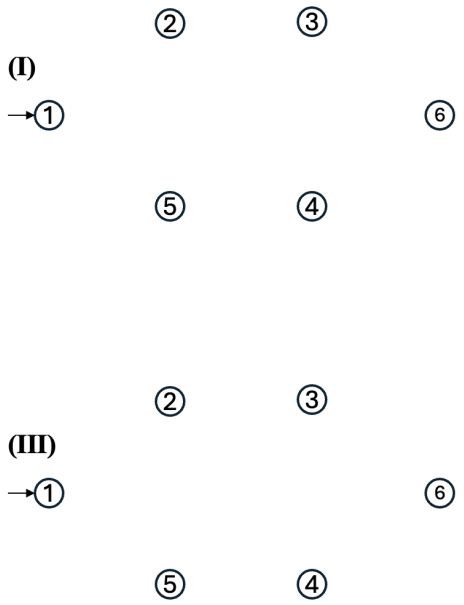
$$\{(1, \lambda, 3), (1, \lambda, 4), (2, b, 3), (3, a, 2), (3, a, 6), (3, \lambda, 6), (4, b, 5), (4, b, 6), (4, \lambda, 6), (5, a, 4)\}.$$



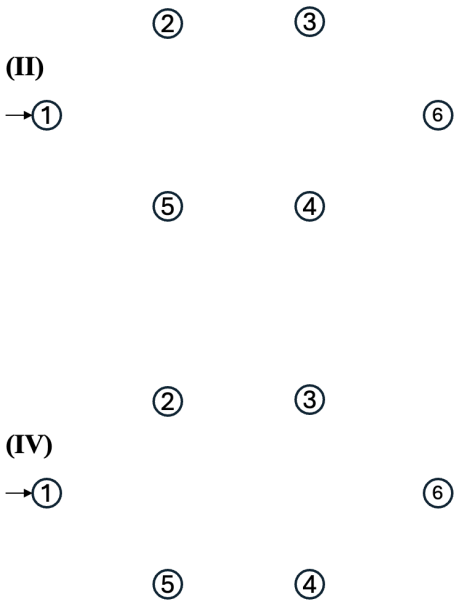
(a, 10) Killing λ-moves: *Using the construction* from the lectures and the textbook, find an ordinary NFA ON that is equivalent to the λ-NFA LN.

- In Graph (I), draw all the edges that are contained in the transitive closure of the λ-edges (including the original λ-edges). Label them with λ.
- In Graph (II), draw all the a-letter moves for ON. Label them as a.
- In Graph (III), draw all the b-letter moves for ON. Label them as b.
- In Graph (IV) draw all the edges in ON properly labeled. Also properly denote the final states of ON.

(I)



(II)



(III)

(IV)

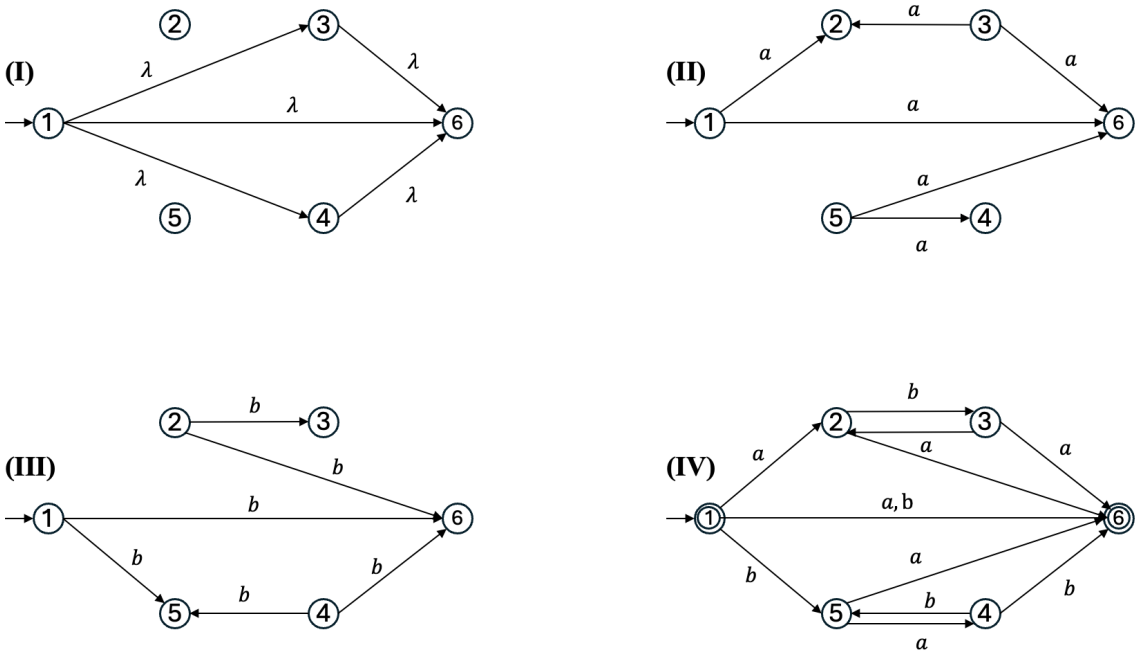
Solution:

We need to add one λ-move $(1, \lambda, 6)$, which is produced by two different pairs of existing λ-moves. Since there is a λ-path from 1 to a final state, we need to make it final in ON.

Each of the six letter moves in LN induces exactly one new letter move along with itself:

- $(2, a, b)$ **induces** $(2, b, 6)$
- $(3, a, 2)$ **induces** $(1, a, 2)$
- $(3, a, 6)$ **induces** $(1, a, 6)$
- $(4, b, 5)$ **induces** $(1, b, 5)$
- $(4, b, 6)$ **induces** $(1, b, 6)$
- $(5, a, 4)$ **induces** $(5, a, 6)$

This gives us a new ordinary NFA ON with twelve letter moves, with final state set $\{1, 6\}$.



Grading notes: *The mean score was 8.3/10 with a median of 9.25. 37% got full marks, with 60% getting 8 0 or above.*

The most common errors were making sates 3 and 4 final on part (IV) (see the marking note below for more detail.

Marking Note 4(a): Only states 1 and 6 were supposed to be final in ON in (IV). A common error was that some students also made states 3 and 4 final.

While it is true that making states 3 and/or 4 final would not change the language accepted by the NFA, making them final was still incorrect.

This is because the problem explicitly said that you should use the method *taught in class and the lectures*. That method would, explicitly, NOT make 3 or 4 final, so we deducted a (very) small penalty for getting this wrong.

At a higher level, the method taught in class was a deterministic algorithm that ALWAYS PROVABLY identifies a correct set of final states. It's rule was that a state will be final in the normal NFA if and only if (a) it was final in the original λ-NFA or (b) it was the start state in the original λ-NFA and the empty string λ was accepted by the original λ-NFA.

In this problem, states 3 and 4 do not satify (a) or (b) so neither of them whould be made final by the algorithm taught in class. For this particular NFA, due to its very specific structure, 3 and 4 *could* also be made final without changing the language accepted but justifying that fact would require extra input-specific work.

(b, 10) Subset Construction: *Using the Subset Construction*, find a DFA D that is equivalent to the NFA ON. To avoid cascading errors in later sections, *we are going to give you* a DFA below (in the statement for Question 4(d)) that will be equivalent to D. You will be graded on whether you carry out the construction on ON to create your own DFA. (Thus you should name your new states according to the construction.) If you are using the construction correctly, you don't need to explain how you get each step of it.

Solution:

We begin with the final start state $\{1\}$**.**

Then $\delta(\{1\}, a)$ **is a new final state** $\{2, 6\}$**.**

Then $\delta(\{1\}, b)$ **is a new final state** $\{5, 6\}$**.**

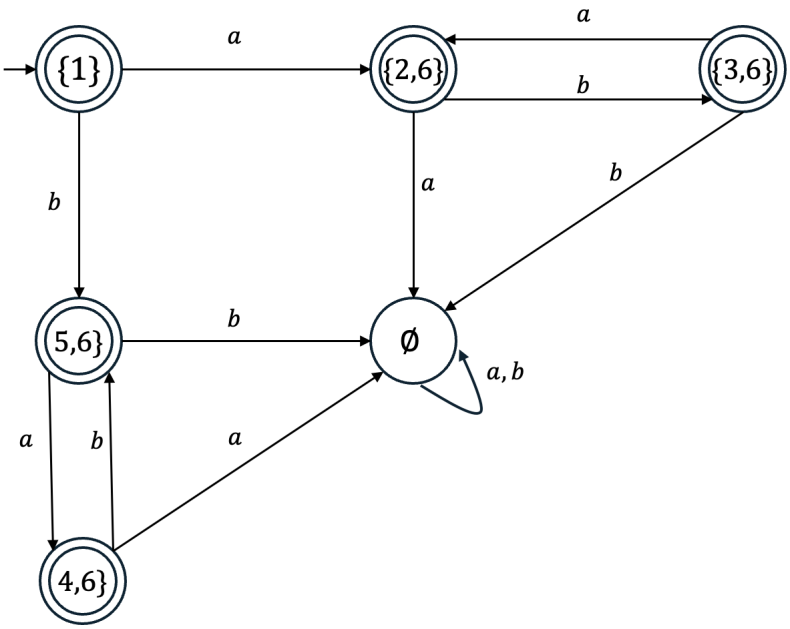
Then $\delta(\{2, 6\}, a)$ **and** $\delta(\{5, 6\})$ **are both the (non-final) death state** $\emptyset$**. As usual,** $\delta(\emptyset, a)$ **and** $\delta(\emptyset, b)$ **are equal to** $\emptyset$**.**

Then $\delta(\{2, 6\}, b)$ **is a new final state** $\{3, 6\}$**.**

Then $\delta(\{5, 6\}, a)$ **is a new final state** $\{4, 6\}$**.**

Then $\delta(\{3, 6\}, a)$ **and** $\{4, 6\}$ **are** $\emptyset$**.**

Then $\delta(\{3, 6\}, a)$ **is** $\{2, 6\}$**, and** $\delta(\{4, 6\}, b)$ **is** $\{5, 6\}$**. Since all arrows to all the states we have created go to existing states, we are done, with a six-state DFA, five of the states being final.**



Grading note: *The mean score was 7.54/10, with 39% getting full marks and a median of 9/10. The rubric indicates specific errors for the start state not being marked, or the death state's arrows being left off. The most common larger error was mishandling the death state, and I took at least three points if your alleged DFA failed to be a DFA (usually by having missing arrows).*

(c, 10) Minimization: Find a DFA MD that is minimal and has the same language as your DFA D from part (b). If you do not use the Minimization Construction, then prove that your new DFA is minimal and that $L(MD) = L(D)$.

Solution:

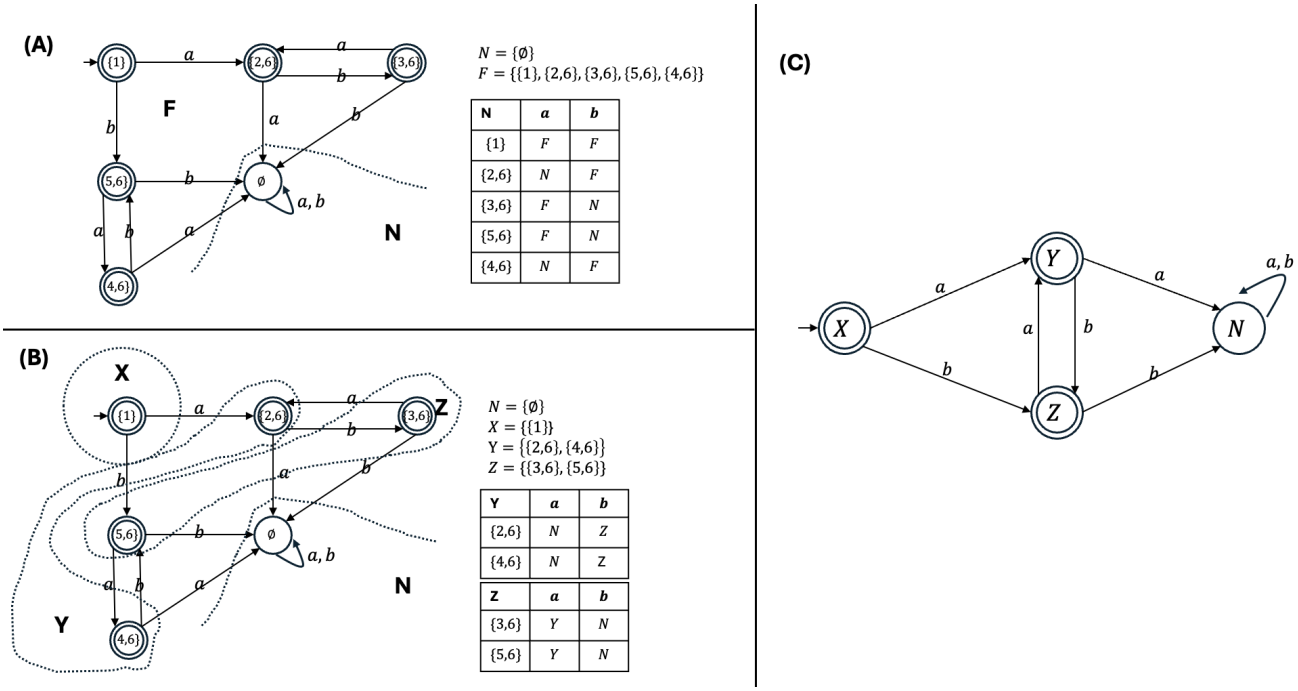
Our first partition has $F = \{1, 26, 36, 46, 56\}$ and $N = \{\emptyset\}$.

(See A) We find that state 1 has behavior FF, states 26 and 46 have behavior NF, and states 36 and 56 have behavior FN.

(See B) So we split class F into $X = \{1\}$, $Y = \{26, 46\}$, and $Z = \{36, 56\}$. With this new partition, states 26 and 46 each have behavior NZ, and states 36 and 56 each have behavior YN.

(See C) So we are done with four states X (start state, final), Y (final), Z (final), and N (non-final). The transitions are:

$$\delta(X, a) = Y), \delta(X, b) = Z, \delta(Y, a) = N$$
$$\delta(Y, b) = Z), \delta(Z, a) = Y, \delta(Z, b) = N$$
$$\delta(N, a) = N, \delta(N, b) = N$$



N	a	b
{1}	F	F
{2,6}	N	F
{3,6}	F	N
{5,6}	F	N
{4,6}	N	F

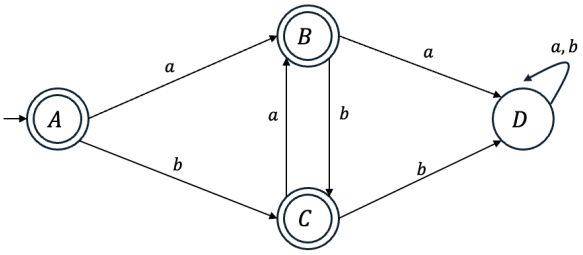
Y	a	b
{2,6}	N	Z
{4,6}	N	Z

Z	a	b
{3,6}	Y	N
{5,6}	Y	N

Grading note: *The mean score was 6.68/10, and 40% got full marks. Many of those full-credit answers correctly minimized an incorrect DFA from Q4b, and I had to deal with a wide variety of such answers. If your start state was non-final here, you only got a penalty here if you had it final in Q4b, since otherwise you would have paid for the error there. In general I did not give more than 4/10 if you were trying to minimize something other than a DFA.*

Family Name: _____

(d, 10) State Elimination: Find and justify a regular expression that is equivalent to the four-state DFA given here, which will be isomorphic to your minimized DFA MD if you solved the earlier parts correctly:

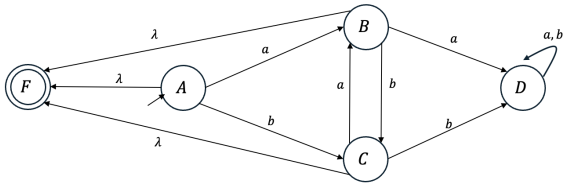


State set $\{A, B, C, D\}$, start state A, final state set $\{A, B, C\}$, transitions $\delta(A, a) = B$, $\delta(A, b) = C$, $\delta(B, a) = D$. $\delta(B, b) = C$, $\delta(C, a) = B$, $\delta(C, b) = \delta(D, a) = \delta(D, b) = D$.

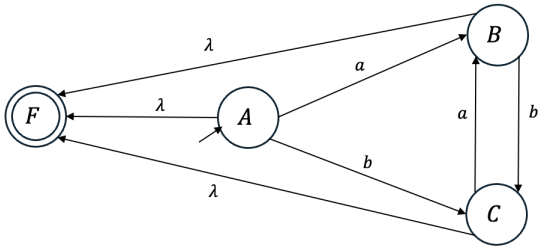
If you use State Elimination on this DFA, no further correctness proof is required. If you use another method, prove that your regular expression is equivalent.

Solution:

We don't need a new start state, so we keep A, B, C, and D and add a new final state F with moves (A, λ, F), (B, λ, F), and (C, λ, F).

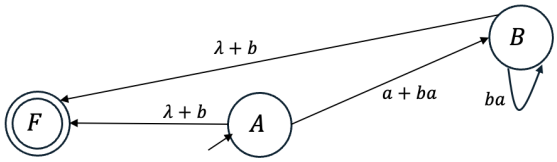


Since D has no arrows out of it, we can eliminate it with no other action.



Either choice of B or C to eliminate first is fine.

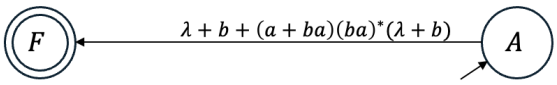
If we eliminate C, we create $(A, \lambda + b, F)$, $(A, a + ba, B)$, (B, ba, B), and $(B, \lambda + b, F)$. (In three of these cases, the newly formed edge merges with an existing parallel edge.)



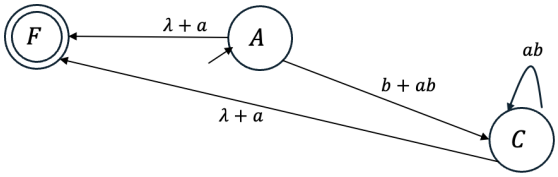
If we then eliminate B, the remaining regular expression is:

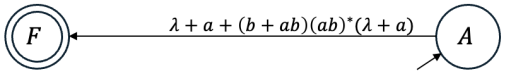
$$\lambda + b + (a + ba)(ba)^*(\lambda + b)$$

.



If we eliminate C first instead and then B we get





and a similar regular expression:

$$\lambda + a + (b + ab)(ab)^*(\lambda + a)$$

.

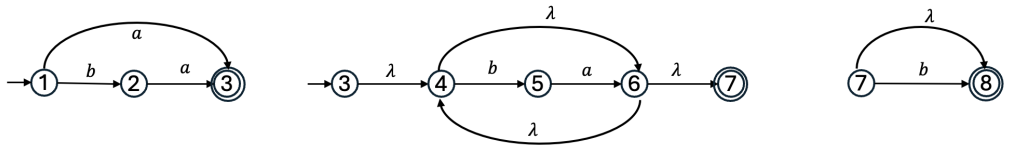
Grading note: *The mean was only 5.49/10, with only 14% getting full marks, even though you were given exactly the DFA to work with. Most of you got the correct setup of new λ-moves, but many of you could have done better to be systematic about the consequences of the last two eliminations. There were a lot of 4/10 answers that were significantly better than the "fragmentary" 2/10 category, and these included anyone who kept part or all of the death state in their final λ-NFA. As we said above, there were a few valid approaches. A lot of the 7/10 answers missed one or more of the consequences of eliminating state B or C, and in most cases I was able to tell you which one in the comment.*

(e, 5XC) Making a λ-NFA: *Using the construction* from lecture and the textbook, compute a λ-NFA from the regular expression in part (d). A different equivalent λ-NFA will get only partial credit. (Though you may omit redundant λ-moves in the construction if you are *sure* that you are not changing the language.)

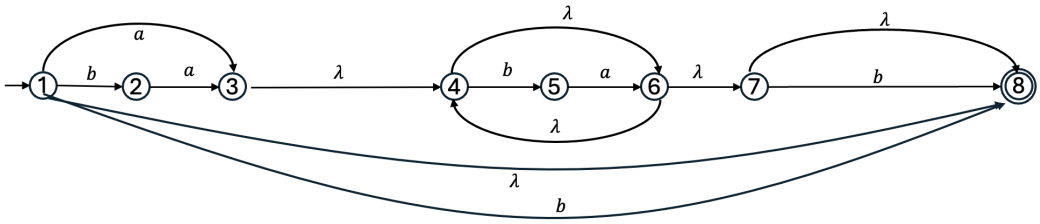
Solution:

We'll do the first of the two regular expressions in the solution to Question 3(d). This expression is the sum of three terms, λ, b, and $(a + ba)(ba)^*)(\lambda + b)$.

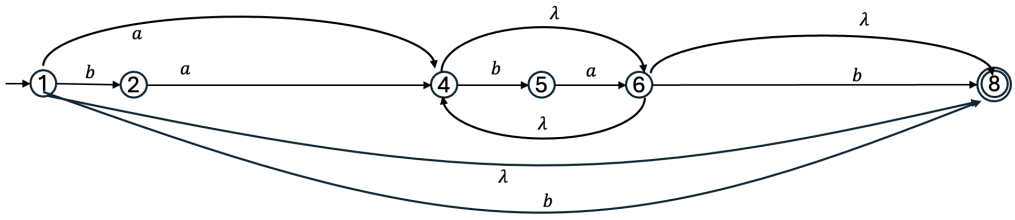
Starting this bottom-up, and following the construction strictly, $(ba)^*$ has a five-state λ-NFA which I will number as states 3 through 7, with transitions $(3, \lambda, 4)$, $(4, b, 5)$, $(4, \lambda, 6)$, $(5, a, 6)$, $(6, \lambda, 4)$ and $(6, \lambda, 7)$. The expression $(a + ba)$ is represented by a three-state λ-NFA which I will number as states 1 through 3, with transitions $(1, b, 2)$, $(1, a, 3)$, and $(2, a, 3)$. The expression $\lambda + b$ is represented by a two-state λ-NFA, which I will number as states 7 and 8, with transitions $(7, \lambda, 8)$ and $(7, b, 8)$.



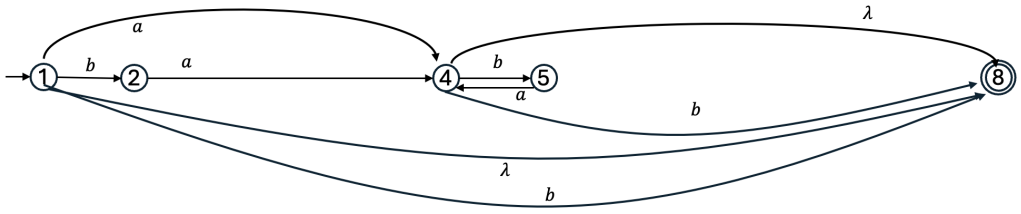
Concatenating these three λ-NFAs, using the given numbering, gives us an eight-state λ-NFA for the result. We make the final λ-NFA by making 1 the start state, making 8 the only final state, and adding two more transitions $(1, \lambda, 8)$ and $(1, b, 8)$.



A reasonable simplification of the λ-NFA for $(ba)^*$ is to merge states 3 and 4 (by removing edge $(3, \lambda, 4)$) and also states 6 and 7 (by removing edge $(6, \lambda, 7)$as follows, giving an equivalent 6 state DFA.



Not as obvious, but easily confirmed, is that the new states states 4 and 6 can now be merged to get the equivalent



on states $\{1, 2, 4, 5, 8\}$.

Grading note: *The mean was 2.47/5, with 22% getting full marks and 32% not attempting it all. I took off a half-point if you left out inessential λ-moves, or put in extra ones. I had a 3/5 category for people who correctly built an expression from their wrong answer from Q4d, if that left them a much easier problem.*

Family Name: _____

Question 5 (20): The following are ten true/false questions, with no explanation needed or wanted, no partial credit for wrong answers, and no penalty for guessing.

After reading the questions, write the correct answer, either T (for true) or F (for false), in the corresponding column. Be sure that your "T" and "F" characters are consistent and distinct.

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)
F	T	T	T	T	F	F	F	T	T

Grading note: These were worse than the last few final exam T/F sections, with an overall mean of 11.79/20, and only one score of 20/20.

- (a) The six statements in the dog proof (Question 1) together provide enough information to determine whether Blaze ate Pie.

  FALSE (89% correct). We know that Blaze exactly ate the same foods as Rhonda and Gwen, and that these include Turkey but not GBC. But nothing in the statements tell us whether the dogs all ate Pie or none did, or for that matter whether they all ate Dog Food, or none did.

- (b) Let "likes" be a symmetric binary relation on dogs. Then it is possible that every dog likes at least one other dog, but there is no dog that likes all other dogs.

  TRUE (65% correct). This certainly happens if there are four dogs, A likes The first statement is $forallx : \exists y : L(x, y)$, and the second is $\exists x : \forall y : L(X, y)$, and the first statement does not imply the second.

- (c) Let Q and R each be equivalence relations on the same set S. Assume for elements a, b, c, and d in S, the pairs (a, b) and (c, d) are in Q, and the pair (b, c) is in R. Then it is possible that (a, d) is not in Q.

  TRUE (61% correct). Let Q be the parity on $S = \{1, 2, 3, 4\}$ and let R be the universal relation on S. Then we may take $a = 1$, $b = 3$, $c = 2$, and $d = 4$, and we have a counterexample.

- (d) Consider the equivalence classes of the integers taken modulo 11. Then it is not true that every class has a multiplicative inverse.

  TRUE (41% correct). The class of 0 does not have an inverse.

- (e) If a and b are distinct nodes of a directed acyclic graph G, and there is a path in G from a to b, then there is no path in G from b to a.

  TRUE (84% correct). If there were such a path, there would be a non-trivial path from a to b and then back to a, which would be a directed cycle, but G is assumed to be acyclic.

- (f) Consider a finite game between two players A and B, represented by a game tree with value v. Then whatever strategy A uses, and whatever strategy B uses, the reward at the end of the game will always be v.

  FALSE (54% correct). Our theorem tells us only that either player *has* a strategy to achieve at worst v for them, but there could certainly other strategies in which a player could fare worse.

- (g) Let Z be the set of all strings w over $\Sigma = \{a, b\}$ such that the number of b's in w is either divisible by 2 or divisible by 3, or both. Then the language of the regular expression $(a^*ba^*ba^*)^* + (a^*ba^*ba^*ba^*)^*$ is exactly Z.

  FALSE (35% correct). This expression gives all strings with a number of b's that is a *positive* multiple of either 2 or 3. The string a, for example, is in Z but not in the language of this expression.

- (h) Let $\Sigma = \{a, b\}$ and let X be the language Σ^*a. Then the languages X and X^R have the same number of Myhill-Nerode equivalence classes. (Recall that X^R is the set of strings whose reversals are in X.)

  FALSE (39% correct). The minimal DFA for X has two states, one final and non-final. The minimal DFA for $X^R = a\Sigma^*$ has three, its start state, a final state with both arrows to itself, and death state. The strings λ, a, and b are pairwise distinguishable.

- (i) There exists a language that can be decided by a Turing machine, but is not the language of any two-way deterministic finite automaton.

  TRUE (56% correct). We sketched a proof in Lecture 37 that two-way DFA's can be simulated by ordinary DFA's, and hence their languages are regular. But we also saw that TM's can decide non-regular languages like the parenthesis language.

- (j) Let X be any language. Then there exists a three-tape nondeterministic Turing machine M such that $X = L(M)$ if and only if X is Turing recognizable.

  TRUE (67% correct). We proved in lecture that multitape Turing machines and nondeterministic Turing machines can each be simulated on an ordinary single-tape deterministic Turing machine, so the language of such a machine must be TR. And any TR language is the language of an ordinary TM, which is a special case of one of these machines.