HW1 : Regular Expressions and Finite Automata

CSE105Sp23

Due: April 11th at 5pm (no penalty late submission until 8am next morning), via Gradescope

In this assignment,

You will practice reading and applying the definitions of alphabets, strings, languages, Kleene star, and regular expressions. You will use regular expressions and relate them to languages and finite automata. You will use precise notation to formally define the state diagram of finite automata, and you will use clear English to describe computations of finite automata informally.

Resources: To review the topics for this assignment, see the class material from Week 1. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Sipser Section 0, 1.3, 1.1. Chapter 1 exercises 1.1, 1.2, 1.3, 1.18, 1.23.

For all HW assignments: Weekly homework may be done individually or in groups of up to 3 students. You may switch HW partners for different HW assignments. The lowest HW score will not be included in your overall HW average. Please ensure your name(s) and PID(s) are clearly visible on the first page of your homework submission and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member(s) to the Gradescope submission by selecting their name(s) in the "Add Group Members" dialog box. You will need to re-add your group member(s) every time you resubmit a new version of your assignment. Each homework question will be graded either for correctness (including clear and precise explanations and justifications of all answers) or fair effort completeness. You may only collaborate on HW with CSE 105 students in your group; if your group has questions about a HW problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) on Piazza.

All submitted homework for this class must be typed. You can use a word processing editor if you like (Microsoft Word, Open Office, Notepad, Vim, Google Docs, etc.) but you might find it useful to take this opportunity to learn LaTeX. LaTeX is a markup language used widely in computer science and mathematics. The homework assignments are typed using LaTeX and you

can use the source files as templates for typesetting your solutions. To generate state diagrams of machines, we recommend using Flap.js or JFLAP. Photographs of clearly hand-drawn diagrams may also be used. We recommend that you submit early drafts to Gradescope so that in case of any technical difficulties, at least some of your work is present. You may update your submission as many times as you'd like up to the deadline.

Integrity reminders

- Problems should be solved together, not divided up between the partners. The homework is designed to give you practice with the main concepts and techniques of the course, while getting to know and learn from your classmates.
- You may not collaborate on homework with anyone other than your group members. You may ask questions about the homework in office hours (of the instructor, TAs, and/or tutors) and on Piazza (as private notes viewable only to the Instructors). You *cannot* use any online resources about the course content other than the class material from this quarter – this is primarily to ensure that we all use consistent notation and definitions (aligned with the textbook) and also to protect the learning experience you will have when the 'aha' moments of solving the problem authentically happen.
- Do not share written solutions or partial solutions for homework with other students in the class who are not in your group. Doing so would dilute their learning experience and detract from their success in the class.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw1CSE105Sp23".

Assigned questions

1. **Functions over sets of strings** (17 points):
For this question, fix the alphabets $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, 2\}$.

Whenever K is a set of strings over Γ and L is a set of strings over Σ, we can use the following rules to define associated sets of strings:

$$\begin{aligned} \textsc{Substring}(K) &:= \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\} \\ \textsc{Rep}(L) &:= \{w \in \Gamma^* \mid \text{between every pair of successive 2s in } w \text{ is a string in } L\} \\ &= \{w \in \Gamma^* \mid \text{for all } v \in \Sigma^* \text{ if } 2v2 \in \textsc{Substring}(\{w\}), \text{ then } v \in L\} \end{aligned}$$

Note: Formally, Substring and Rep are functions whose domains and codomains are specified as

$$\textsc{Substring} : \mathcal{P}(\Gamma^*) \to \mathcal{P}(\Gamma^*)$$

and

$$\textsc{Rep} : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Gamma^*)$$

In other words, SUBSTRING maps sets of strings with characters $\{0, 1, 2\}$ to associated sets of strings with characters $\{0, 1, 2\}$; and REP maps sets of strings with characters in $\{0, 1\}$ to associated sets of strings with characters in $\{0, 1, 2\}$.

(a) (*Graded for correctness*) [1] Consider $w = 0120$ (which is a string in Γ^*). List every element of the set SUBSTRING($\{w\}$). In other words, fill in the blank

$$\text{SUBSTRING}(\{w\}) = \{\_\_\_\_\_\}$$

Briefly justify your answer by referring back to the relevant definitions.

Not graded, but good to think about: Why do we need the curly braces—"{" and "}"—around w for the input to SUBSTRING*?*

(b) (*Graded for correctness*) Specify an example language A over Γ such that $A \neq \Gamma^*$ and yet SUBSTRING(A) $= \Gamma^*$, or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example language A and a precise and clear description of the result of computing SUBSTRING(A) using relevant definitions to justify this description and to justify the set equality with Γ^*, or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

(c) (*Graded for completeness*) [2] Define the language B to be the language over Σ described by the regular expression

$$\Sigma^* 1 \Sigma^*$$

In plain English, we might explain that B is the set of all strings of 0s and 1s that contain a 1. Give a plain English explanation for the set of strings REP(B).

(d) (*Graded for correctness*) Prove/disprove: For every finite language L over Σ, REP(L) is also a finite set of strings. A complete answer will either give a general argument starting with an arbitrary finite language and proving that the result of applying REP is also finite, or will give a counterexample (which is a specific example of a finite language L for which applying REP gives an infinite language, with justification referring back to the relevant definitions).

Note: A finite language is a set of finitely many strings. This includes the possibility that L is the empty set!

(e) (*Graded for completeness*) Write a template for a regular expression that describes REP(L) when L is described by a regular expression R. You may use union, concatenation, Kleene star, and Σ, Γ, and R. (We're using the shorthand for regular expressions describing alphabets from page 64.)

[1] This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

[2] This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we ask that you include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

2. **Deciphering regular expressions** (22 points):
For this question, let's fix the regular expression over the alphabet $\{0, 1\}$

$$R = 0^*(1 \cup 10)^*$$

For each choice of strings of length 3, $a, b, c \in \{0, 1\}^3$ we can define the regular expression:

$$X_{a,b,c} = 0(a \cup b \cup c)^*$$

(a) (*Graded for completeness*) Give a plain English explanation for the language described by the regular expression R. This continues a theme from Problem 1—before trying to prove formal statements about a specific regular expression, it's often good to try to translate it into a form that is more easy to reason about. Typically speaking, the shorter and more concise your plain English description is, the more useful it will be in reasoning about the language.

(b) (*Graded for correctness*) Suppose $a = 000$, $b = 001$, $c = 011$ so

$$X_{a,b,c} = 0(000 \cup 001 \cup 011)^*$$

Show that $L(R) \not\subseteq L(X_{a,b,c})$ by giving some string in $L(R)$ which is not in $L(X_{a,b,c})$, and justifying this choice referring back to relevant definitions.

(c) (*Graded for correctness*) More generally, prove that

$$L(R) \not\subseteq L(X_{a,b,c})$$

for *all* possible strings $a, b, c \in \{0, 1\}^3$. Hint: What are the possible lengths of strings in $L(R)$ (and why does this help)?

(d) (*Graded for correctness*) Give a specific example of three distinct strings $a, b, c \in \{0, 1, 2\}^3$ such that

$$L(X_{a,b,c}) \subseteq L(R)$$

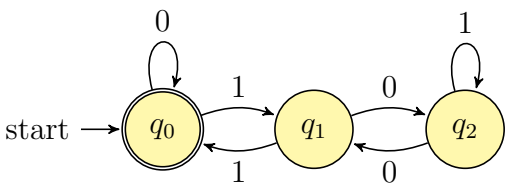
Briefly justify your answer by explaining how an arbitrary element of $L(X_{a,b,c})$ is guaranteed to be an element of $L(R)$.

(e) (*Graded for correctness*) Give a specific example of three distinct strings $a, b, c \in \{0, 1, 2\}^3$ such that

$$L(X_{a,b,c}) \not\subseteq L(R)$$

Briefly justify your answer by giving a counterexample string that is in $L(X_{a,b,c})$ and is not in $L(R)$ (and explaining why using relevant definitions).

3. **The right transition function can make or break a DFA** (6 points):
Consider the finite automaton $(Q, \Sigma, \delta, q_0, F)$ depicted below



where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, and $F = \{q_0\}$.

(a) (*Graded for completeness*) Find and fix the mistake in the following symbolic description of the transition function $\delta\colon Q \times \Sigma \to Q$: for each $j \in \{0, 1\}$

$$\delta(q_0, j) = q_j \qquad \delta(q_1, j) = q_{1-j} \qquad \delta(q_2, j) = q_{1+j}$$

(b) (*Graded for correctness*) Keeping the same set of states $Q = \{q_0, q_1, q_2\}$, alphabet $\Sigma = \{0, 1\}$, starting state q_0, and set of accepting states $F = \{q_0\}$, change the transition function δ so that the resulting finite automaton recognizes the language described by the regular expression

$$0^* \cup \Sigma^* 1000^*$$

Briefly justify why the resulting finite automaton works by describing the role of each state with your new transition function and relating it to a plain English description of the language described by the regular expression.

Note: with regular expressions $*$ binds more tightly than concatenation so $1000^* = (100)(0^*)$.

(Challenge question, not graded) There is a beautiful plain English description of the language recognized by the finite automaton with the state diagram depicted at the start of Problem 3. What is it?

4. **Being precise with terminology** (5 points):
For each of the following statements, determine if it is true, false, or if the question doesn't even make sense (because the statement isn't well formed or doesn't use terms in ways consistent with definitions from class).

(a) (*Graded for completeness*) The empty string is in every language.

(b) (*Graded for completeness*) Σ^* is a language.

(c) (*Graded for completeness*) Every language is a regular expression.

(d) (*Graded for completeness*) Alphabets are infinite.

(e) (*Graded for completeness*) There is a (finite) number $k \in \mathcal{N}$ such that every DFA has fewer than k states.

HW2 : Regular Languages and Automata Constructions Due: April 18th at 5pm (no penalty late submission until 8am next morning), via Gradescope

You will practice designing multiple representations of regular languages and working with general constructions of automata to demonstrate the richness of the class of regular languages.

Resources: To review the topics you are working with for this assignment, see the class material from Week 1 and Week 2. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Sipser Section 1.1, 1.2, 1.3. Chapter 1 exercises 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.11, 1.12, 1.14, 1.15, 1.16, 1.17, 1.19, 1.20, 1.21, 1.22.

Key Concepts: Regular expressions, language described by a regular expression, deterministic finite automata (DFAs), regular languages, closure of the class of regular languages under certain operations, nondeterministic finite automata (NFA).

For all HW assignments: Weekly homework may be done individually or in groups of up to 3 students. You may switch HW partners for different HW assignments. The lowest HW score will not be included in your overall HW average. Please ensure your name(s) and PID(s) are clearly visible on the first page of your homework submission and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member(s) to the Gradescope submission by selecting their name(s) in the "Add Group Members" dialog box. You will need to re-add your group member(s) every time you resubmit a new version of your assignment. Each homework question will be graded either for correctness (including clear and precise explanations and justifications of all answers) or fair effort completeness. You may only collaborate on HW with CSE 105 students in your group; if your group has questions about a HW problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) on Piazza.

All submitted homework for this class must be typed. You can use a word processing editor if you like (Microsoft Word, Open Office, Notepad, Vim, Google Docs, etc.) but you might find it useful to take this opportunity to learn LaTeX. LaTeX is a markup language used widely in computer science and mathematics. The homework assignments are typed using LaTeX and you can use the source files as templates for typesetting your solutions. To generate state diagrams of machines, we recommend using Flap.js or JFLAP. Photographs of clearly hand-drawn diagrams may also be used. We recommend that you submit early drafts to Gradescope so that in case of any technical difficulties, at least some of your work is present. You may update your submission as many times as you'd like up to the deadline.

Integrity reminders

- Problems should be solved together, not divided up between the partners. The homework is designed to give you practice with the main concepts and techniques of the course, while

getting to know and learn from your classmates.

- You may not collaborate on homework with anyone other than your group members. You may ask questions about the homework in office hours (of the instructor, TAs, and/or tutors) and on Piazza (as private notes viewable only to the Instructors). You *cannot* use any online resources about the course content other than the class material from this quarter – this is primarily to ensure that we all use consistent notation and definitions (aligned with the textbook) and also to protect the learning experience you will have when the 'aha' moments of solving the problem authentically happen.
- Do not share written solutions or partial solutions for homework with other students in the class who are not in your group. Doing so would dilute their learning experience and detract from their success in the class.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw2CSE105Sp23".

Assigned questions

1. **It can be hard to give a good complement** (15 points):
   For any language $L \subseteq \Sigma^*$, recall that we define its *complement* as

   $$\overline{L} := \Sigma^* - L = \{w \in \Sigma^* \mid w \notin L\}$$

   That is, the complement of L contains all and only those strings which are not in L. Our notation for regular expressions does not include the complement symbol. However, it turns out that the complement of a language described by a regular expression is guaranteed to also be describable by a (different) regular expression. For example, over the alphabet $\Sigma = \{0, 1\}$, the complement of the language described by the regular expression Σ^*0 is described by the regular expression $\varepsilon \cup \Sigma^*1$ because any string that does not end in 0 must either be the empty string or end in 1.

   For each of the regular expressions R over the alphabet $\Sigma = \{0, 1\}$ below, write the regular expression for $\overline{L(R)}$. Your regular expressions may use the symbols $\varnothing$, ε, 0, 1, and the following operations to combine them: union, concatenation, and Kleene star.

   Briefly justify why your solution for each part works by giving plain English descriptions of the language described by the regular expression and of its complement and connecting them to the regular expression via relevant definitions. An English description that is more detailed than simply negating the description in the original language will likely be helpful in the justification.

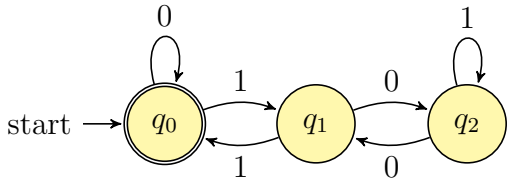
   (a) (*Graded for correctness*) [3] $(\Sigma\Sigma)^*$

[3] This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

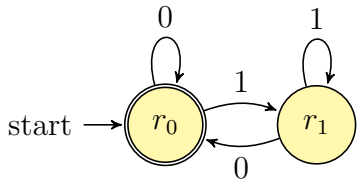
(b) (*Graded for correctness*) $\Sigma^*11\Sigma^*$

(c) (*Graded for correctness*) $0^*10^*10^*$

2. **Closure of the class of regular languages under intersection** (12 points):
For this question, let $\Sigma = \{0, 1\}$. Recall the DFA over Σ from the previous homework:



We'll call the language recognized by the DFA above A. Let's also define a new language $B \subseteq \Sigma^*$ to be the language recognized by the DFA over Σ with state diagram below:



(a) (*Graded for correctness*) Using the construction for the intersection of two regular languages (Sipser page 46), draw the state diagram for a DFA recognizing the intersection of the languages A and B. The labels of each one of your states should be the ordered pair of labels for the states from the two machines above. Your diagram should have 6 states.

(b) (*Graded for completeness*) In this part of the problem, you will prove that the general construction for the DFA recognizing intersection of two languages that you used in part (a) does not always produce a DFA with the smallest number of states possible. You will do this by giving one counterexample (that combined with your work in part (a), proves the general claim). Your task: design a DFA with exactly 4 states that recognizes the language $A \cap B$. Briefly justify why your design works by describing the role of each state of your DFA and relating it to a plain English description of the language resulting from the intersection.

(c) (*Graded for correctness*) Later in the class we will learn that there are some languages which are not regular, and in fact, we will learn specific techniques to prove that certain languages are not regular. For the moment, however, we can already investigate closure properties of the class of regular languages just by knowing that a non-regular language exists.

We know (from the textbook and our work in class) that if L and K are regular languages, then $L \cap K$ is regular (for arbitrary languages L and K). Prove that the converse of this statement is false; that is, give a counterexample by giving a specific

regular language L so that for each non-regular language X, $L \cap X$ is regular (even though X isn't).

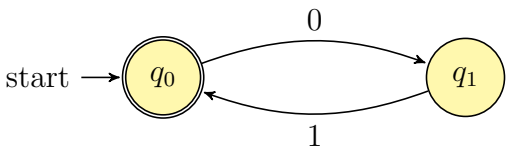
In your solution, justify why L is regular and why $L \cap X$ is regular (for arbitrary X) using relevant definitions.

(Challenge question, not graded) Prove/disprove: For any language L over Σ^*, $L \cap B$ is regular implies L is regular, where B is the specific language from part (a) and (b) of Problem 2.

3. **Closure of the class of regular languages under** SUBSTRING (16 points):
Let $\Gamma = \{0, 1, 2\}$. From the previous homework, recall the function SUBSTRING that has domain and codomain $\mathcal{P}(\Gamma^*)$, where, for each language K over Γ,

$$\text{SUBSTRING}(K) := \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\}$$

(a) (*Graded for correctness*) Consider the NFA over Γ with state diagram:



We'll call the language recognized by the NFA above C. Fill in the blanks below:

- An example of a string over Γ that is in C **and** is in SUBSTRING(C) is _____ because _____
- An example of a string over Γ that is in C **and** is **not** in SUBSTRING(C) is _____ because _____
- An example of a string over Γ that is **not** in C **and** is in SUBSTRING(C) is _____ because _____
- An example of a string over Γ that is **not** in C **and** is **not** in SUBSTRING(C) is _____ because _____

For each item, you'll either fill in a specific string and a justification that refers back to the relevant definitions, or you'll write "impossible" for the first part of the sentence and justify why it's impossible to find such an example referring back to the relevant definitions.

(b) (*Graded for completeness*) Prove that the class of regular languages is closed under the SUBSTRING operation. Namely, give a general construction that takes an arbitrary NFA and constructs an NFA that recognizes the result of applying SUBSTRING to the language recognized by the original machine. You can describe your construction in words and/or draw a picture to illustrate your construction. You do not have to write down a formal specification.

(c) (*Graded for completeness*) Draw the state diagram of an NFA over Γ that recognizes SUBSTRING(C) (for C the language from part (a) of this Problem), using your construction from part (b) of this Problem, or manually constructing it. Describe the computation(s) of this NFA for each of the sample strings you gave in part (a).

4. **Closure of the class of regular star-free languages under** REP (7 points):
A language is said to be *star-free* whenever it can be described by a regular expression that has no Kleene star operations, but where complement operation can be incorporated into the expression as many times as you like. For example, the language

$$\{\varepsilon, 0010\}$$

is star-free because it can be described by $\varepsilon \cup 0010$ which does not use the Kleene star operation symbol.

(a) (*Graded for correctness*) Prove that the set of all strings over $\Gamma = \{0, 1, 2\}$ is star-free. A complete solution will give an expression that describes this language that does not use Kleene star but may incoporate the complement expression as many times as you like, along with a justification that refers back to relevant definitions.

(b) (*Graded for completeness*) Prove that every finite language is star-free.

(c) (*Graded for completeness*) Let $\Sigma = \{0, 1\}$. From the previous homework, recall the function REP that has domain $\mathcal{P}(\Sigma^*)$ and codomain $\mathcal{P}(\Gamma^*)$, where, for each language L over Σ,

$$\text{REP}(L) := \{w \in \Gamma^* \mid \text{between every pair of successive 2's in } w \text{ is a string in } L\}$$

Show that $\text{REP}(L)$ is a regular and star-free language whenever L is a regular and star-free language. That is, given an expression R describing L, write a regular expression for $\text{REP}(L)$ using only the regular expressions R, $\varnothing$, ε, 0, 1, 2, and the following operations to combine them: union, concatenation, and complement. You may assume that $\overline{R}$ describes $\Sigma^* - L(R)$, that is, the complement for the regular expression R over the alphabet Σ is itself a language over Σ.

HW3 : Nonregular Languages and Pushdown Automata Due: April 25th at 5pm (no penalty late submission until 8am next morning), via Gradescope

In this assignment:

You will practice distinguishing between regular and nonregular languages using both closure arguments and the pumping lemma.

Resources: To review the topics you are working with for this assignment, see the class material from Week 2 through Week 4. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Sipser Section 1.4, 2.2. Chapter 1 exercises 1.29, 1.30. Chapter 1 problems 1.49, 1.50, 1.51.

Key Concepts: Pumping lemma, pumping length, regular languages, nonregular languages, pushdown automata, stack.

For all HW assignments: Weekly homework may be done individually or in groups of up to 3 students. You may switch HW partners for different HW assignments. The lowest HW score will not be included in your overall HW average. Please ensure your name(s) and PID(s) are clearly visible on the first page of your homework submission and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member(s) to the Gradescope submission by selecting their name(s) in the "Add Group Members" dialog box. You will need to re-add your group member(s) every time you resubmit a new version of your assignment. Each homework question will be graded either for correctness (including clear and precise explanations and justifications of all answers) or fair effort completeness. You may only collaborate on HW with CSE 105 students in your group; if your group has questions about a HW problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) on Piazza.

All submitted homework for this class must be typed. You can use a word processing editor if you like (Microsoft Word, Open Office, Notepad, Vim, Google Docs, etc.) but you might find it useful to take this opportunity to learn LaTeX. LaTeX is a markup language used widely in computer science and mathematics. The homework assignments are typed using LaTeX and you can use the source files as templates for typesetting your solutions. To generate state diagrams of machines, we recommend using Flap.js or JFLAP. Photographs of clearly hand-drawn diagrams may also be used. We recommend that you submit early drafts to Gradescope so that in case of any technical difficulties, at least some of your work is present. You may update your submission as many times as you'd like up to the deadline.

Integrity reminders

- Problems should be solved together, not divided up between the partners. The homework

is designed to give you practice with the main concepts and techniques of the course, while getting to know and learn from your classmates.

- You may not collaborate on homework with anyone other than your group members. You may ask questions about the homework in office hours (of the instructor, TAs, and/or tutors) and on Piazza (as private notes viewable only to the Instructors). You *cannot* use any online resources about the course content other than the class material from this quarter – this is primarily to ensure that we all use consistent notation and definitions (aligned with the textbook) and also to protect the learning experience you will have when the 'aha' moments of solving the problem authentically happen.
- Do not share written solutions or partial solutions for homework with other students in the class who are not in your group. Doing so would dilute their learning experience and detract from their success in the class.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw3CSE105Sp23".

Requests from your TAs and tutors To help us with grading please

- Start each question on a new page.
- Label the start of each solution with **Answer**.

Assigned questions

1. **Regular or not?** (21 points):
   Fix $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, 2\}$. For each of the languages listed below, prove that it is either regular or nonregular. *Note:* You might find it useful to explore the definition of each set and consider alternate (simpler) ways of stating it.

   For each language that is regular, a complete solution will include a precise definition of a DFA, NFA, or regular expression that recognizes or describes it, along with a brief justification of your construction by explaining the role each state plays in the machine or referring back to relevant definitions.

   For each language that is nonregular, a complete solution will use the pumping lemma to prove that it is nonregular, including appropriate justification related to the specific language.

   (a) (*Graded for correctness*) [4] $L_1 = \{0^n x 1^n \mid n \geq 1, x \in \Sigma^*\}$, a language over Σ.

[4] This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

(b) (*Graded for correctness*) $L_2 = \{0^n 1 x 0 1^n \mid n \geq 1, x \in \Sigma^*\}$, a language over Σ.

(c) (*Graded for correctness*) Recall that for $L \subseteq \Sigma^*$, we define

$$\begin{aligned}\textsc{Rep}(L) &:= \{w \in \Gamma^* \mid \text{between every pair of successive 2's in } w \text{ is a string in } L\} \\ &= \{w \in \Gamma^* \mid \text{for all } v \in \Sigma^* \text{ if } 2v2 \in \textsc{Substring}(\{w\}), \text{ then } v \in L\}\end{aligned}$$

$L_3 = \textsc{Rep}(\{0^n 1^n \mid n \geq 1\})$, a language over Γ.

2. **Properties of nonregular languages** (21 points):
Prove or disprove each of the following statements. In other words, decide whether each statement is true or false and justify your decision. Let $\Sigma = \{0, 1\}$ and let $\Gamma = \{0, 1, 2\}$.

(a) (*Graded for correctness*) For all languages L, K over Σ, if L is nonregular and K is finite, then $L - K$ is nonregular. Recall: $L - K = \{w \in \Sigma^* \mid w \in L \text{ and } w \notin K\}$.

(b) (*Graded for correctness*) Every infinite language over Σ where each string in the language has an equal number of 0's and 1's is nonregular.

(c) (*Graded for correctness*) Recall that for language K over Γ,

$$\textsc{Substring}(K) := \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\}.$$

For every nonregular language K over Γ, $\textsc{Substring}(K)$ is nonregular.

3. **Pumping dilemma** (8 points):
Your friend claims that the Pumping Lemma is useless for proving that an infinite language $K \subseteq \Sigma^*$ is not regular. Their logic goes like this

(Step 1) Suppose that K is regular. It can be recognized by a DFA $M = (Q, \Sigma, \delta, q_0, F)$.

(Step 2) For arbitrary DFA M, the pumping length p is at least $|Q|$.

(Step 3) However, for every integer $n \geq |Q|$, there exists a machine $M' = (Q', \Sigma, \delta', q_0', F')$ such that $L(M') = L(M) = K$ and $|Q'| = n$.

(Step 4) Therefore, the Pumping Lemma cannot be used to pump any string of finite length since its pumping length might be arbitrarily large.

Below, we will examine the steps above in detail. Justify your answer to each part.

(a) (*Graded for completeness*) [5] (Step 1): Is this statement true? In other words, just because we're assuming that K is regular a regular language, does it mean we can assume there is a DFA that recognizes it?

[5] This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we ask that you include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

(b) (*Graded for completeness*) (Step 2): In general, it's true that the smallest the pumping length of a language recognized by a DFA with states Q can be is $|Q|$. Prove this by finding a specific infinite language K and a DFA recognizing where K cannot have pumping length smaller than $|Q|$.

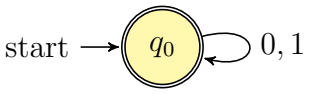
(c) (*Graded for completeness*) (Step 3): This step is correct; prove the stated version of this statement: For every integer $n \geq |Q|$, there exists a machine $M' = (Q', \Sigma, \delta', q_0', F')$ such that $L(M') = L(M)$ and $|Q'| = n$.

(*Challenge; not graded*): Define a *cycle* to be a sequence of *distinct* states $q_1, q_2, \ldots, q_m$ such that

$$\delta(q_1, \sigma_1) = q_2, \qquad \delta(q_2, \sigma_2) = q_3, \quad \ldots, \quad \delta(q_m, \sigma_m) = q_1,$$

where $\sigma_1, \sigma_2, \ldots, \sigma_m \in \Sigma$ are symbols in the alphabet. An objection to the statement in (Step 3) is that the proof of the Pumping Lemma depends on the length of the cycles in the DFA rather than the number of states. That is, increasing the number of states in your DFA might not increase the pumping length because the length of the smallest cycle stays the same. Nevertheless, a version of your friend's statement is still true whenever you impose this additional cycle constraint: for every integer $n \geq |Q|$, there exists a machine $M' = (Q', \Sigma, \delta', q_0', F')$ such that $L(M') = L(M)$ and the length of the smallest cycle in the M' is at least n.

Your task is to show that even this more general statement is true for the simple language Σ^* recognized by the DFA below:



For all $n \geq 1$, define a DFA for this language where the length of the smallest cycle is n.

(d) (*Graded for completeness*) (Step 4): Describe why this statement is true/false/misleading.

HW4 : Pushdown Automata and Context-free grammars Due: May 2nd at 5pm (no penalty late submission until 8am next morning), via Gradescope

In this assignment:

You will practice with the definition of pushdown automata and context-free grammars and reason about regular and context-free languages.

Resources: To review the topics you are working with for this assignment, see the class material from Week 3 through Week 4. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Sipser Sections 2.1, 2.2. Chapter 2 exercises 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.9, 2.10, 2.11, 2.12, 2.13, 2.16, 2.17.

Key Concepts: Pushdown automata, stack, context-free grammars, derivations, context-free languages.

For all HW assignments: Weekly homework may be done individually or in groups of up to 3 students. You may switch HW partners for different HW assignments. The lowest HW score will not be included in your overall HW average. Please ensure your name(s) and PID(s) are clearly visible on the first page of your homework submission and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member(s) to the Gradescope submission by selecting their name(s) in the "Add Group Members" dialog box. You will need to re-add your group member(s) every time you resubmit a new version of your assignment. Each homework question will be graded either for correctness (including clear and precise explanations and justifications of all answers) or fair effort completeness. You may only collaborate on HW with CSE 105 students in your group; if your group has questions about a HW problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) on Piazza.

All submitted homework for this class must be typed. You can use a word processing editor if you like (Microsoft Word, Open Office, Notepad, Vim, Google Docs, etc.) but you might find it useful to take this opportunity to learn LaTeX. LaTeX is a markup language used widely in computer science and mathematics. The homework assignments are typed using LaTeX and you can use the source files as templates for typesetting your solutions. To generate state diagrams of machines, we recommend using Flap.js or JFLAP. Photographs of clearly hand-drawn diagrams may also be used. We recommend that you submit early drafts to Gradescope so that in case of any technical difficulties, at least some of your work is present. You may update your submission as many times as you'd like up to the deadline.

Integrity reminders

- Problems should be solved together, not divided up between the partners. The homework

is designed to give you practice with the main concepts and techniques of the course, while getting to know and learn from your classmates.

- You may not collaborate on homework with anyone other than your group members. You may ask questions about the homework in office hours (of the instructor, TAs, and/or tutors) and on Piazza (as private notes viewable only to the Instructors). You *cannot* use any online resources about the course content other than the class material from this quarter – this is primarily to ensure that we all use consistent notation and definitions (aligned with the textbook) and also to protect the learning experience you will have when the 'aha' moments of solving the problem authentically happen.
- Do not share written solutions or partial solutions for homework with other students in the class who are not in your group. Doing so would dilute their learning experience and detract from their success in the class.

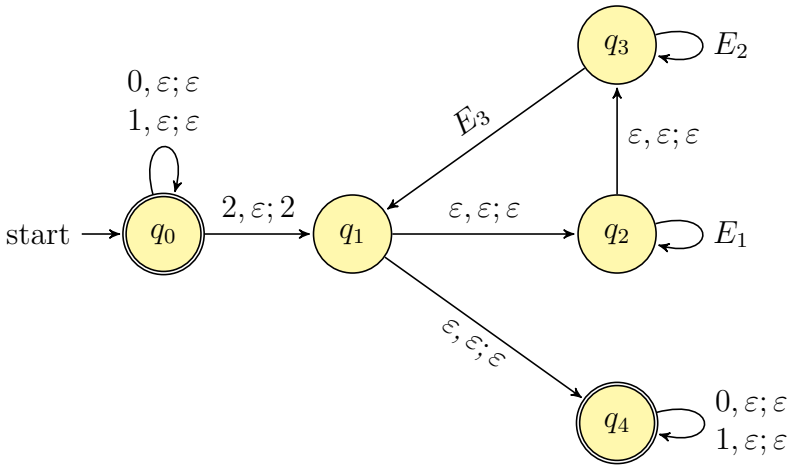
You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw4CSE105Sp23".

Requests from your TAs and tutors To help us with grading please

- Start each question on a new page.
- Label the start of each solution with **Answer**.

Assigned questions

1. **A PDA with multiple possibilities** (22 points):
   Consider the PDA with input and stack alphabet $\Gamma = \{0, 1, 2\}$ whose "unfinished" state diagram is given below:



There are three labels (E_1, E_2, and E_3) on the edges that are unspecified. To be precise, each E_i is of the form "$x, y; z$" where $x, y, z \in \Gamma_\varepsilon$ (recall $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$).

(a) (*Graded for correctness*) [6] Prove that (no matter how the labels E_1, E_2, E_3 are specified), the language recognized by this PDA is infinite. A complete solution will include a precise description of an infinite collection of strings each of which is accepted by the PDA, with a precise and clear description of the accepting computation of the PDA on each of these strings.

(b) (*Graded for completeness*) [7] Prove/Disprove: Over all the possible choices for the labels E_1, E_2, E_3, this PDA can only recognize finitely many languages. Justify your solution by referring back to the relevant definitions.

(c) (*Graded for correctness*) Recall that for $L \subseteq \Sigma^*$ with $\Sigma = \{0, 1\}$, we define

$$\begin{aligned}\textsc{Rep}(L) &:= \{w \in \Gamma^* \mid \text{between every pair of successive 2's in } w \text{ is a string in } L\} \\ &= \{w \in \Gamma^* \mid \text{for all } v \in \Sigma^* \text{ if } 2v2 \in \textsc{Substring}(\{w\}), \text{ then } v \in L\}\end{aligned}$$

where for all languages $K \subseteq \Gamma^*$ we let

$$\textsc{Substring}(K) := \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\}.$$

Determine how to set the labels E_1, E_2, E_3 so that the language of the PDA is

$$\textsc{Rep}(\{0^n 1^m \mid n \geq 0, m \geq 0\})$$

In addition to specifying each E_i, a complete justification will include a precise description of why this choice of the E_i's means that the PDA recognizes the language indicated.

(d) (*Graded for correctness*) Determine how to set the labels E_1, E_2, E_3 so that the language of the PDA is

$$\textsc{Rep}(\{0^n 1^n \mid n \geq 0\})$$

In addition to specifying each E_i, a complete justification will include a precise description of why this choice of the E_i's means that the PDA recognizes the language indicated.

[6] This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

[7] This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we ask that you include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

2. **Grammar practice** (12 points):
   For each of the languages listed below, define a context-free grammar $G = (V, \Sigma, R, S)$ that generates the language. Instead of formally justifying your grammar, illustrate it by giving **two examples** of strings in the language and their derivations using your grammar and **one example** of a string not in the language with an explanation of why it cannot appear on the right side of any derivation in your grammar. Choose your examples so they are different enough to illustrate the role of as many of the variables in your grammar as possible.

   (a) (*Graded for correctness*) $\textsc{Rep}(\{0^n1^n \mid n \geq 0\})$

   (b) (*Graded for correctness*) $\{1^n = 1^a + 1^b \in \{1, =, +\}^* \mid a, b, n \geq 1 \text{ such that } a + b = n\}$

3. **Substrings and regularity** (16 points):
   For this problem, we fix the alphabet $\Gamma = \{0, 1, 2\}$. Recall the definition of the function Substring from Problem 1.

   (a) (*Graded for correctness*) Prove that $\textsc{Substring}(\{0^n1^n \mid n \geq 0\})$ is regular. A complete solution will include a precise definition of a DFA, NFA, or regular expression that recognizes or describes it, along with a brief justification of your construction by explaining the role each state plays in the machine and referring back to relevant definitions.

   (b) (*Graded for correctness*) Prove that $\textsc{Substring}(\{0^n1^n2^n \mid n \geq 0\})$ is not regular.

   (c) (*Graded for completeness*) Is $\textsc{Substring}(\{0^n1^n2^n \mid n \geq 0\})$ context-free? Informally justify your answer, referring to class discussions and/or the textbook.

HW5 : Turing Machines Due: May 16th at 5pm (no penalty late submission until 8am next morning), via Gradescope

In this assignment: You will practice designing and working with Turing machines and their variants.

Resources: To review the topics you are working with for this assignment, see the class material from Week 5 through Week 6. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Chapter 3. Chapter 3 exercises 3.1, 3.2.

Key Concepts: Formal definitions of Turing machines, computations of Turing machines, halting computations, implementation-level descriptions of Turing machines, high-level descriptions of Turing machines, recognizable languages, decidable languages, variants of Turing machines, enumerators, nondeterministic Turing machines, Church-Turing thesis.

For all HW assignments: Weekly homework may be done individually or in groups of up to 3 students. You may switch HW partners for different HW assignments. The lowest HW score will not be included in your overall HW average. Please ensure your name(s) and PID(s) are clearly visible on the first page of your homework submission and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member(s) to the Gradescope submission by selecting their name(s) in the "Add Group Members" dialog box. You will need to re-add your group member(s) every time you resubmit a new version of your assignment. Each homework question will be graded either for correctness (including clear and precise explanations and justifications of all answers) or fair effort completeness. You may only collaborate on HW with CSE 105 students in your group; if your group has questions about a HW problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) on Piazza.

All submitted homework for this class must be typed. You can use a word processing editor if you like (Microsoft Word, Open Office, Notepad, Vim, Google Docs, etc.) but you might find it useful to take this opportunity to learn LaTeX. LaTeX is a markup language used widely in computer science and mathematics. The homework assignments are typed using LaTeX and you can use the source files as templates for typesetting your solutions. To generate state diagrams of machines, we recommend using Flap.js or JFLAP. Photographs of clearly hand-drawn diagrams may also be used. We recommend that you submit early drafts to Gradescope so that in case of any technical difficulties, at least some of your work is present. You may update your submission as many times as you'd like up to the deadline.

Integrity reminders

- Problems should be solved together, not divided up between the partners. The homework is designed to give you practice with the main concepts and techniques of the course, while

getting to know and learn from your classmates.

- You may not collaborate on homework with anyone other than your group members. You may ask questions about the homework in office hours (of the instructor, TAs, and/or tutors) and on Piazza (as private notes viewable only to the Instructors). You *cannot* use any online resources about the course content other than the class material from this quarter – this is primarily to ensure that we all use consistent notation and definitions (aligned with the textbook) and also to protect the learning experience you will have when the 'aha' moments of solving the problem authentically happen.
- Do not share written solutions or partial solutions for homework with other students in the class who are not in your group. Doing so would dilute their learning experience and detract from their success in the class.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw5CSE105Sp23".

Assigned questions

1. REP(**Describing a Turing Machine for** REP) (18 points):
   Recall that for $L \subseteq \Sigma^*$ with $\Sigma = \{0, 1\}$, we define the language over $\Gamma = \{0, 1, 2\}$

   $$\begin{aligned}\textsc{Rep}(L) &:= \{w \in \Gamma^* \mid \text{between every pair of successive 2's in } w \text{ is a string in } L\} \\ &= \{w \in \Gamma^* \mid \text{for all } v \in \Sigma^* \text{ if } 2v2 \in \textsc{Substring}(\{w\}), \text{ then } v \in L\}\end{aligned}$$

   where for all languages $K \subseteq \Gamma^*$ we let

   $$\textsc{Substring}(K) := \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\}.$$

   In this question, you will give three separate descriptions of a Turing machine which recognizes the language $\textsc{Rep}(\{0^n 1^n \mid n \geq 0\})$.

   This may seem like a somewhat tedious process, but we think that it is important to see all the different descriptions in action at least once for a single language.

   With each description, give a brief justification connecting the description to the language it recognizes.

   (a) (*Graded for completeness*) [8] High-level description: description of algorithm (precise sequence of instructions), without implementation details of machine. Your description can use data structures and refer to specific positions in the input strings (without specifying memory management).

[8] This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we ask that you include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

(b) (*Graded for correctness*) [9] Implementation-level description: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.

(c) (*Graded for correctness*) Formal definition: Give the 7-tuple of parameters

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

describing the Turing Machine. Represent the transition function δ by drawing the state diagram of the Turing machine. You may use the following conventions: omit the reject state from the diagram; any missing transitions in the state diagram are assumed to go to the reject state.

2. **This Turing Machine is broken... or is it?** (12 points):
Let's consider a variant of a Turing Machine which models computation where the data keeps getting corrupted. A Corrupted Turing Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ has some of the usual Turing Machine components: set of states (Q); input alphabet (Σ); transition function (δ); start state (q_0); accepting state (q_{acc}), rejecting state (q_{rej}). Unlike a normal Turing Machine, the tape alphabet $\Gamma \supseteq \Sigma$ has two (rather than one) special characters $\sqcup$ and $\lightning$:

- $\sqcup \in \Gamma$, $\sqcup \notin \Sigma$: This is the usual blank symbol.
- $\lightning \in \Gamma$, $\lightning \notin \Sigma$: This symbol indicates a cell that has been corrupted. The machine cannot write over any corrupted cell; that is, for each $q \in Q$,

$$\delta(q, \lightning) \in \{(r, \lightning, D) \mid r \in Q, D \in \{L, R\}\}$$

Computation in the corrupted Turing Machine proceeds as normal except that sometimes when writing a tape symbol as intended, a $\lightning$ symbol is written instead. Thankfully, the pattern of corruption is predictable: the first write is corrupted, and then every other write thereafter is corrupted. That is, the first, third, fifth,... etc. writes are corrupted.

(a) (*Graded for correctness*) Prove that for every **regular** language L, there exists a corrupted Turing Machine M that recognizes L.

(b) (*Graded for completeness*) It will turn out that the Corrupted Turing Machine is no less powerful than our usual definition of a Turing Machine. Let's break the proof into a few steps. First, define the alphabet $\Sigma_{\text{pairs}} := \Sigma \times \Sigma = \{(a, b) \mid a \in \Sigma, b \in \Sigma\}$ of pairs of symbols in Σ. Give the construction of a Corrupted Turing Machine that takes input $a = a_1 a_2 \cdots a_n \in \Sigma^*$ and rewrites it as pairs of symbols interspersed by

[9] This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

corrupted symbols. To be precise, the Corrupted Turing Machine should take starting tape configuration

a_1	a_2	$\ldots$	a_{n-1}	a_n	$\sqcup$	$\sqcup$	$\ldots$

to the tape configuration

$\lightning$	(a_1, a_2)	$\lightning$	(a_3, a_4)	$\lightning$	$\ldots$	$\lightning$	(a_{n-1}, a_n)	$\lightning$	$\sqcup$	$\sqcup$	$\ldots$

with the head once again pointing to the first cell. For simplicity, you may assume that n is even.

(c) (*Graded for completeness*) Starting from the tape configuration we created in the previous step, describe the implementation of a Corrupted Turing Machine which simulates the computation of any normal/uncorrupted Turing Machine.

Challenge; not graded: A key idea in the construction above was to increase the number of symbols in our tape alphabet. Can you do the same construction *without* increasing the size of the tape alphabet? That is, for every language L recognized by a normal Turing machine over alphabet Σ and tape alphabet $\Sigma \cup \{\sqcup\}$, is there a Corrupted Turing Machine with tape alphabet $\Sigma \cup \{\sqcup, \lightning\}$ that recognizes L?

3. **True/False enumerator** (20 points):
For each of the following statements, determine if it is true or false. Clearly label your choice by starting your solution with **True** or **False** and then provide a brief (3-4 sentences or so) justification for your answer.

(a) (*Graded for correctness*) Every enumerator enumerates an infinite language.

(b) (*Graded for correctness*) Let E be any enumerator and M be any Turing machine. If $L(E) = L(M)$, then M enters the reject state for all strings not in $L(E)$.

(c) (*Graded for correctness*) Let E be any enumerator over Σ. Suppose $a, b \in \Sigma^*$ and $a, b \in L(E)$. If E prints a before b, then $|a| \leq |b|$.

(d) (*Graded for correctness*) Let M decide language L over Σ such that M halts on all inputs $w \in \Sigma^*$ in $|w|^{2023}$ steps. There exists an enumerator E with the following properties: $L(E) = L(M)$; and if $a, b \in L$ and $|a| < |b|$, then E prints a before b.

(e) (*Graded for correctness*) Let N be a nondeterministic Turing machine. There is an enumerator E that enumerates the set of all and only strings accepted by N that have odd length.

HW6 : Computational Problems, Recognizability, Decidability Due: May 23th at 5pm (no penalty late submission until 8am next morning), via Gradescope

In this assignment: You will use general constructions and specific machines to explore the classes of recognizable and decidable languages. You will explore various ways to encode machines as strings so that computational problems can be recognized.

Resources: To review the topics you are working with for this assignment, see the class material from Week 6 through Week 7. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Chapter 4 exercises 4.1, 4.3, 4.4., 4.5. Chapter 4 Problems 4.29, 4.30.

Key Concepts: uring-recognizable languages, Turing-decidable languages, Church-Turing thesis, computational problems.

For all HW assignments: Weekly homework may be done individually or in groups of up to 3 students. You may switch HW partners for different HW assignments. The lowest HW score will not be included in your overall HW average. Please ensure your name(s) and PID(s) are clearly visible on the first page of your homework submission and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member(s) to the Gradescope submission by selecting their name(s) in the "Add Group Members" dialog box. You will need to re-add your group member(s) every time you resubmit a new version of your assignment. Each homework question will be graded either for correctness (including clear and precise explanations and justifications of all answers) or fair effort completeness. You may only collaborate on HW with CSE 105 students in your group; if your group has questions about a HW problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) on Piazza.

All submitted homework for this class must be typed. You can use a word processing editor if you like (Microsoft Word, Open Office, Notepad, Vim, Google Docs, etc.) but you might find it useful to take this opportunity to learn LaTeX. LaTeX is a markup language used widely in computer science and mathematics. The homework assignments are typed using LaTeX and you can use the source files as templates for typesetting your solutions. To generate state diagrams of machines, we recommend using Flap.js or JFLAP. Photographs of clearly hand-drawn diagrams may also be used. We recommend that you submit early drafts to Gradescope so that in case of any technical difficulties, at least some of your work is present. You may update your submission as many times as you'd like up to the deadline.

Integrity reminders

- Problems should be solved together, not divided up between the partners. The homework is designed to give you practice with the main concepts and techniques of the course, while

getting to know and learn from your classmates.

- You may not collaborate on homework with anyone other than your group members. You may ask questions about the homework in office hours (of the instructor, TAs, and/or tutors) and on Piazza (as private notes viewable only to the Instructors). You *cannot* use any online resources about the course content other than the class material from this quarter – this is primarily to ensure that we all use consistent notation and definitions (aligned with the textbook) and also to protect the learning experience you will have when the 'aha' moments of solving the problem authentically happen.
- Do not share written solutions or partial solutions for homework with other students in the class who are not in your group. Doing so would dilute their learning experience and detract from their success in the class.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw6CSE105Sp23".

Assigned questions

1. **Explicit encodings** (8 points):
   In a computational problem, the elements of the language are encodings of machines. For example, consider the language

   $$E_{\text{DFA}} := \{\langle M \rangle \mid M \text{ is a DFA, and } L(M) = \varnothing\}$$

   where each string $\langle M \rangle$ in the language encodes a DFA $M = (Q, \Sigma, \delta, q_0, F)$. Usually, we purposefully drop the details about how this encoding is done because they can distract from the central computational properties of the language. In fact, any encoding can be used so long as there exists a decider for syntactic questions about the DFAs being encoded. In this question, we will build some specific explicit examples of encodings of DFAs to get more comfortable with these ideas.

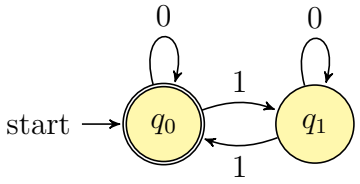
   (a) (*Graded for completeness*) [10] *Encoding with delimiters:* Perhaps the most straightforward way to create an encoding is to have it mirror the structure of the tuple $(Q, \Sigma, \delta, q_0, F)$ for the DFA. Your task: describe an encoding that maps each DFA M to a distinct string $\langle M \rangle$ that uniquely identifies M. That is, if you "decode" the encoding, you get the exact same machine back.

   Hints, tips, notes of caution:

[10] This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we ask that you include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

- You may use special characters like # and $ as delimiters in your encoding to separate the various components.
- Your encoding alphabet must be finite

(b) (*Graded for completeness*) Use your encoding from part (a) to produce the string encoding the DFA below:



(c) (*Graded for completeness*) Show that it is possible to have the same kind of delimited encoding without using special delimiter characters. In particular, prove that for every DFA M, we can assume that $\langle M \rangle \subseteq \{0,1\}^*$.

Challenge; not graded:
For the delimited encoding schemes above, there are strings over the encoding alphabet (Σ) that nevertheless do not correspond to a valid DFA.

Prove/disprove: There exists an encoding scheme for which this is not true; that is,

$$\{\langle M \rangle \mid M \text{ is a DFA}\} = \Sigma^*.$$

2. **Closure** (18 points):
Let $\Sigma = \{0,1\}$ and $\Gamma = \{0,1,2\}$. Recall the functions

$$\begin{aligned} \textsc{Substring}(K) &:= \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\} \\ \textsc{Rep}(L) &:= \{w \in \Gamma^* \mid \text{between every pair of successive 2s in } w \text{ is a string in } L\} \\ &= \{w \in \Gamma^* \mid \text{for all } v \in \Sigma^* \text{ if } 2v2 \in \textsc{Substring}(\{w\}), \text{ then } v \in L\} \end{aligned}$$

(a) (*Graded for correctness*) [11] Prove that, given any deterministic decider over Σ, M_L, there is a deterministic decider over Γ that recognizes

$$\textsc{Rep}(\ L(M_L)\ )$$

In other words, you will prove that for any Turing-decidable language L over Σ, $\textsc{Rep}(L)$ is also Turing-decidable. A complete answer will include both a precise construction of the machine and a (brief) justification of why this machine works as required.

(b) (*Graded for correctness*) Prove that, given any nondeterministic Turing machine over Γ, N_L, there is a nondeterministic Turing machine over Γ that recognizes

$$\textsc{Substring}(\ L(N_L)\ )$$

In other words, you will prove that the class of Turing-recognizable languages over Γ is closed under the SUBSTRING operation. A complete answer will include both a precise construction of the machine and a (brief) justification of why this machine works as required.

(c) (*Graded for completeness*) Give a different proof that the class of Turing-recognizable languages over Γ is closed under the SUBSTRING operation, this time using only deterministic Turing machines. A complete answer will include both a precise construction of the machine and a (brief) justification of why this machine works as required.

3. **Computational problems** (24 points):
For each of the following statements, determine if it is true or false. Clearly label your choice by starting your solution with **True** or **False** and then provide a brief (3-4 sentences or so) justification for your answer.

(a) (*Graded for correctness*) For each regular language K, the language

$$\{\langle M\rangle \mid M \text{ is a DFA and } L(M) = K\}$$

is decidable.

(b) (*Graded for correctness*) For each regular language L, the language

$$\{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are both DFA and } L(M_1) \subseteq L \text{ and } L(M_2) \subseteq \overline{L}\}$$

is decidable.

(c) (*Graded for correctness*) Let Model $\in \{\text{DFA}, \text{NFA}, \text{REX}, \text{CFG}, \text{PDA}\}$. If EQ_{Model} is decidable, then E_{Model} is decidable.

Challenge; not graded: Let Model $\in \{\text{DFA}, \text{NFA}, \text{REX}, \text{CFG}, \text{PDA}\}$. *If* A_{Model} *is decidable, then* EQ_{Model} *is decidable.*

[11] This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

HW7 : Undecidability, Co-Recognizability, and Mapping Reductions Due: June 6th at 5pm (no penalty late submission until 8am next morning), via Gradescope

In this assignment: You will use general constructions and specific machines to explore the classes of recognizable, decidable, and undecidable languages. You will use computable functions to relate the difficulty levels of languages via mapping reduction.

Resources: To review the topics you are working with for this assignment, see the class material from Week 7 through Week 9. We will post frequently asked questions and our answers to them in a pinned Piazza post.

Reading and extra practice problems: Chapter 5 exercises 5.4, 5.5, 5.6, 5.7. Chapter 5 problems 5.10, 5.11, 5.16, 5.18.

Key Concepts: Computational problems, diagonalization, undecidability, unrecognizability, computable function, mapping reduction.

For all HW assignments: Weekly homework may be done individually or in groups of up to 3 students. You may switch HW partners for different HW assignments. The lowest HW score will not be included in your overall HW average. Please ensure your name(s) and PID(s) are clearly visible on the first page of your homework submission and then upload the PDF to Gradescope. If working in a group, submit only one submission per group: one partner uploads the submission through their Gradescope account and then adds the other group member(s) to the Gradescope submission by selecting their name(s) in the "Add Group Members" dialog box. You will need to re-add your group member(s) every time you resubmit a new version of your assignment. Each homework question will be graded either for correctness (including clear and precise explanations and justifications of all answers) or fair effort completeness. You may only collaborate on HW with CSE 105 students in your group; if your group has questions about a HW problem, you may ask in drop-in help hours or post a private post (visible only to the Instructors) on Piazza.

All submitted homework for this class must be typed. You can use a word processing editor if you like (Microsoft Word, Open Office, Notepad, Vim, Google Docs, etc.) but you might find it useful to take this opportunity to learn LaTeX. LaTeX is a markup language used widely in computer science and mathematics. The homework assignments are typed using LaTeX and you can use the source files as templates for typesetting your solutions. To generate state diagrams of machines, we recommend using Flap.js or JFLAP. Photographs of clearly hand-drawn diagrams may also be used. We recommend that you submit early drafts to Gradescope so that in case of any technical difficulties, at least some of your work is present. You may update your submission as many times as you'd like up to the deadline.

Integrity reminders

- Problems should be solved together, not divided up between the partners. The homework is designed to give you practice with the main concepts and techniques of the course, while

getting to know and learn from your classmates.

- You may not collaborate on homework with anyone other than your group members. You may ask questions about the homework in office hours (of the instructor, TAs, and/or tutors) and on Piazza (as private notes viewable only to the Instructors). You *cannot* use any online resources about the course content other than the class material from this quarter – this is primarily to ensure that we all use consistent notation and definitions (aligned with the textbook) and also to protect the learning experience you will have when the 'aha' moments of solving the problem authentically happen.
- Do not share written solutions or partial solutions for homework with other students in the class who are not in your group. Doing so would dilute their learning experience and detract from their success in the class.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw7CSE105Sp23".

Assigned questions

1. **Properties of mapping reductions** (20 points):
   In the review quizzes, we saw that mapping reductions are transitive and are not symmetric. That is, if $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$ and there are sets A, B where $A \leq_m B$ but it is not the case that $B \leq_m A$.

   In this question, we'll explore other properties of mapping reductions. We fix the alphabet Σ and all sets we consider are languages over this alphabet.

   For each of the following statements, determine if it is true or false. Clearly label your choice by starting your solution with **True** or **False** and then provide a brief (3-4 sentences or so) justification for your answer.

   (a) (*Graded for correctness*) [12] Mapping reductions are *not* related to subset inclusion. That is, there are example sets A, B, C, D where $A \subseteq B$ and $A \leq_m B$ and $C \not\subseteq D$ and $C \leq_m D$. *Note: the notation $C \not\subseteq D$ means that C is not a subset of D. That is, there is an element of C that is not an element of D.*

   (b) (*Graded for correctness*) For every decidable language L, there is a regular language R such that $L \leq_m R$.

   (c) (*Graded for correctness*) Mapping reducibility is preserved under complement. That is, for all sets A and B, if $A \leq_m B$, then $\overline{A} \leq_m \overline{B}$.

[12] This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

(d) (*Graded for correctness*) $A \leq_m B$ for every decidable language A and every co-recognizable language B. *Note: the definition of co-recognizable is from Week 8 and is: A language L over an alphabet Σ is called* **co-recognizable** *if its complement, defined as $\Sigma^* \setminus L = \{x \in \Sigma^* \mid x \notin L\}$, is Turing-recognizable.*

2. **What's wrong with these reductions?** (20 points):
Suppose your friends are practicing coming up with mapping reductions $A \leq_m B$ and their witnessing functions $f : \Sigma^* \to \Sigma^*$. For each of the following attempts, determine if it is has error(s) or is correct. Do so by labelling each attempt with all and only the labels below that apply, and justifying this labelling.

- *Error Type 1:* The given function can't witness the claimed mapping reduction because there exists an $x \in A$ such that $f(x) \notin B$.
- *Error Type 2:* The given function can't witness the claimed mapping reduction because there exists an $x \notin A$ such that $f(x) \in B$.
- *Error Type 3:* The given function can't witness the claimed mapping reduction because the specified function is not computable.
- *Correct:* The claimed mapping reduction is true and is witnessed by the given function.

Clearly present your answer by first listing all the relevant labels from above and then providing a brief (3-4 sentences or so) justification for each of those labels.

(a) (*Graded for correctness*) $A_{\text{TM}} \leq_m HALT_{\text{TM}}$ and

$$f(x) = \begin{cases} \langle \text{start} \to (q_{\text{acc}}) \rangle & \text{if } x = \langle M, w \rangle \text{ for a Turing machine } M \text{ and string } w \\ & \quad \text{and } w \in L(M) \\ \langle \text{start} \to (q_0) \circlearrowright 0, 1, \sqcup \to R \rangle & \text{otherwise} \end{cases}$$

(b) (*Graded for correctness*) $\{ww \mid w \in \{0,1\}^*\} \leq_m \{w \mid w \in \{0,1\}^*\}$ and

$$f(x) = \begin{cases} w & \text{if } x = ww \text{ for a string } w \text{ over } \{0,1\} \\ \varepsilon & \text{otherwise} \end{cases}$$

(c) (*Graded for correctness*) $EQ_{\text{TM}} \leq_m A_{\text{TM}}$ with

$$f(x) = \begin{cases} \langle \text{start} \to (q_{\text{acc}}), M_w \rangle & \text{if } x = \langle M, w \rangle \text{ for a Turing machine } M \text{ and string } w \\ \varepsilon & \text{otherwise.} \end{cases}$$

Where for each Turing machine M, we define

$M_w =$ "On input y
1. Simulate M on w.
2. If it accepts, accept.
3. If it rejects, reject."

You may assume that ε is never a valid encoding and that encodings of pairs of Turing machines are never the same as encodings of a Turing machine and an input string (i.e., $\langle M_1, M_2 \rangle \neq \langle M_3, w \rangle$).

3. **Computational histories** (10 points):
At any point in the computation of a Turing machine, we can record what is going on by (metaphorically) taking a "snapshot". We want this snapshot to contain all the information needed to simulate the rest of the computation. In particular, the snapshot encodes the

   - Tape contents: Although the tape is infinite, at any specific point in a computation, only finitely many cells have been used. These are the only relevant tape contents to be encoded.
   - Head position: An index to which position on the tape the head is currently pointing.
   - State: An index to which state in the finite control the computation is currently at.

   Notice that much like the encoding $\langle M \rangle$ of a Turing machine M, we can encode all of this snapshot information in a single string called a *configuration* (usually denoted by the letter C). In the same spirit of getter functions for components of encodings, all of the relevant information can effectively be extracted from the configuration. More formally, there is a computable function which computes each of the tape contents, head position, and state given a configuration as input. See Sipser Figure 3.4 (and surrounding discussion) for an explicit example of a configuration.

   A computational history for Turing machine M on input w is sequence of configurations $C_1, C_2, \ldots, C_k$ such that configuration C_{i+1} results from taking one step in the Turing machine computation corresponding to C_i (in other words, one application of the transition function). Additionally, C_1 is the starting configuration, corresponding to the tape that has the characters w on the leftmost $|w|$-many cells of the tape, the tape head at the leftmost position, and the current state being the starting state of the Turing machine. We say that a computational history is *accepting* if the final configuration in the sequence C_k has the current state being the accept state of the Turing machine.

   Let's suppose we can describe both the encodings of Turing machines and configurations using the alphabet $\Sigma = \{0, 1\}$. That is, $\langle M \rangle \in \Sigma^*$ and $C \in \Sigma^*$ for any Turing machine M and configuration of the Turing machine C. We define the language of accepting computational histories over the alphabet $\Gamma = \{0, 1, 2\}$:

$$H := \{\langle M \rangle 2 \langle w \rangle 2 C_1 2 \cdots 2 C_k \mid M \text{ is a Turing machine, } w \text{ is a string, } C_1, \ldots, C_k \text{ is the computational history of } M \text{ on } w \text{ and is accepting}\}$$

   That is, strings in the language H start with an encoding of some Turing machine M, followed by an encoding of some string w, followed by an accepting computational history of M on input w. There is a 2 symbol between each of these components to serve as a delimiter. To be clear, each of these encodings is over the alphabet $\{0, 1\}$, but you may

also assume that it's possible to decide whether or not a particular bit string is an encoding of a Turing machine, a configuration, or neither.

(a) (*Graded for correctness*) Give a high-level description for a Turing machine that decides H and justify why it works. Namely, prove that the Turing machine you define halts for each input and that it accepts an arbitrary string if and only if that string is in H.

(b) (*Graded for completeness*) [13] Prove that $\textsc{Substring}(H)$ is undecidable by showing a mapping reduction from A_{TM}. That is, you will prove that $A_{\text{TM}} \leq_m \textsc{Substring}(H)$ by giving a witnessing function. Recall that for any language $K \subseteq \Gamma^*$, we define

$$\textsc{Substring}(K) := \{w \in \Gamma^* \mid \text{there exist } a, b \in \Gamma^* \text{ such that } awb \in K\}.$$

Combining parts (a) and (b), notice that this implies that the class of decidable languages is not closed under the Substring operation.

[13] This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we ask that you include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

Project - CSE 105 Spring 2022 Due 6/1/23 at 5pm (no penalty late submission until 8am next day)

The project component is designed for you to go deeper and extend your work on assignments and to explore an application of your choosing. The project is an individual assignment and has two tasks:

- Task 1: A meaningful language (written) and
- Task 2: A helpful function (some programming, presented as a screencast video).

What resources can you use? This project must be completed individually, without any help from other people, including the course staff (other than logistics support if you get stuck with screencast).

You can use any of this quarter's CSE 105 offering (notes, readings, class videos, homework feedback). These resources should be more than enough.

If you are struggling to get started and want to look elsewhere online, you must acknowledge this by listing and citing any resources you consult (even if you do not explicitly quote them), including any large-language model style resources. Link directly to them and include the name of the author / video creator, any search strings or prompts you used, and the reason you consulted this reference.

The work you submit for the project needs to be your own. Again, you shouldn't need to look anywhere other than this quarter's material and doing so may result in definitions or notations that conflict with our norms in this class so think carefully before you go down this path.

Submitting the project You will submit a PDF for Task 1 and a video for Task 2 to an online assignment on Gradescope.

Task 1: A meaningful language

Automaton models are useful for concisely representing patterns. In this question, you'll choose a pattern in an application you care about, define it precisely, and then build a DFA, NFA, or PDA that recognizes it.

First, pick **one** application for your example. Here are some ideas to get you started - but you can choose to go in a different direction all together.

- Data validation for input in text files (e.g. emails with specific domains, dates in specific formats, PIDs in a class list, etc.)
- Finding ASCII codes for punctuation in a binary file.
- The CDC recommended procedure for hand washing (Refer to the guidelines from the CDC here https://www.cdc.gov/handwashing/index.html in your explanation. You might find the first example in chapter 1 about automatic door controllers helpful when starting your design.)
- See more ideas here:

  `https://theory-cs.github.io/files/practical-applications-of-theory-of-computation.pdf`

Then:

1. In a paragraph or so, give the context for your chosen application and why you chose it.
2. Specify the alphabet for your example.
3. Write a precise (mathematical and/or English) description of a set of strings over this alphabet that is important, and include a sentence or so justifying why this set is important.
4. Give one example of a string in this set and a string not in this set, and explain why you chose these example strings.
5. Classify your language as regular or context-free, and prove this classification by giving an appropriate machine that recognizes your language. Justify your construction.

Grading criteria and checklists Solution is typed out in detail step-by-step, with clear and correct logic and justification.

Each of the five items are included, with precise language and notation for all terms and complete, correct, and clear justification.

Task 2: A helpful function

To relate the difficulty level of one language to another we use mapping reduction, which relies on the notion of computable function. In this part of the project, you will define, program, and trace a specific computable function from $\{a, b\}^*$ to $\{a, b\}^*$.

First, choose a function you will be working with. You can pick any function you like so long as:

- Its domain is $\{a, b\}^*$ and its codomain is $\{a, b\}^*$
- There is at least one domain element that is mapped to a string that is **longer** than it and there is at least one domain element that is mapped to a string that is **shorter** than it.

Then:

1. Give a high-level description of a Turing machine that computes this function.
2. Draw the state diagram of this Turing machine.
3. Write a program in Java, Python, JavaScript, or C++ (or another programming language of your choosing) that simulates the computations of this Turing machine. Your program should display a snapshot of each step of the computation, including the state of the machine, the current (non-blank) contents of the tape, and the location of the read/write head of the Turing machine. If you would like, you may use aids such as co-pilot or ChatGPT to help you write this program. However, you should test the code that is produced and be able to explain what it is doing. As a header in your code file, include a comment block describing any resources that were used to help generate your code.
4. Run your program on a string over $\{a, b\}$ which the function maps to a string that is longer than it, and
5. Run your program on a string over $\{a, b\}$ which the function maps to a string that is shorter than it.

Presenting your reasoning and demonstrating it via screenshare are important skills that also show us a lot of your learning. Getting practice with this style of presentation is a good thing for you to learn in general and a rich way for us to assess your skills. To demonstrate your work, you will create a 3-5 minute screencast video with the following components:

- Start with your face and your student ID for a few seconds at the beginning, and introduce yourself audibly while on screen. You don't have to be on camera for the rest of the video, though it's fine if you are. We are looking for a brief confirmation that it's you creating the video and doing the work you submitted.

- Present the function you will be working with. Your video should include a clear and precise definition of the function (before you introduce any Turing machines).
- Show on the screen and explain the high-level description of your Turing machine witnessing that your function is computable.
- Show on the screen and explain the state diagram of your Turing machine.
- Show on the screen and present your code, including the software design choices you made (e.g. which data structures are you using, etc.) and any resources you used.
- Demonstrate running your code on the two example input strings specified above. Explain why the output of your program is what you would expect.

Grading criteria and checklists Logistics: video loads correctly, is between 3 and 5 minutes, shows the student's face and ID, and they introduce themself audibly while on screen.

The video clearly states which function was chosen for study, the function which is well-defined and computable, the video presents the two different descriptions of the Turing machine clearly, and the Turing machine correctly computes the function.

The video clearly describes which programming language was chosen for the implementation and gives the reasons why.

The video discusses the connections between the state diagram of the Turing machine and its implementation in the code.

The video clearly demonstrates all test cases, including both expected and actual output. The video should include screencasts of running the code live to demonstrate these test cases.

Your video: You may produce screencasts with any software you choose. One option is to record yourself with Zoom; a tutorial on how to use Zoom to record a screencast (courtesy of Prof. Joe Politz) is here:

`https://drive.google.com/open?id=1KROMAQuTCk4OzwrEFotlYSJJQdcG_GUU`.

The video that was produced from that recording session in Zoom is here:

`https://drive.google.com/open?id=1MxJN6CQcXqIbOekDYMxjh7mTt1TyRVMl`

Please send an email to the instructors (minnes@ucsd.edu and dgrier@ucsd.edu) if you have concerns about the video / screencast components of this project or cannot complete projects in this style for some reason.