CSE 311 : Spring 2022 Final Exam Solutions

Name:	**NetID:** @uw.edu

Instructions

- You have one-hour and fifty-minutes to complete this exam.
- You are permitted one piece of 8.5x11 inch paper with handwritten notes (notes are allowed on both sides of the paper).
- You may not use a calculator or any other electronic devices during the exam.
- We will be scanning your exams before grading them. Please write legibly, and avoid writing up to the edge of the paper.
- If you run out of room, indicate (e.g. with an arrow) that you're going to use the back of the sheet, and continue writing there.
- You may also use the last page for extra space, but tell us where to find your answer if it's not right below the problem.

Advice

- Remember to properly format English proofs (e.g. introduce all your variables).
- All proofs for this exam must be English proofs.
- We give partial credit for the beginning and end of a proof. Even if you don't know how the middle goes, you can write the start of the proof and put the "target" and conclusion at the bottom.
- Remember to take deep breaths.

Question	**Max points**
Training Wheels	14
First Proof	11
Models of Computation	12
Induction I	20
Short Answer	12
Induction II	20
Wait, That's Illegal	15
Grading Morale	1
Total	**105**

1. Training Wheels [14 points]

Let your domain of discourse be positive integers. Use the following predicates:

- $\texttt{Prime}(x)$ which is true if and only if x is prime.
- $\texttt{Divides}(x, y)$ which is true if and only if $x|y$.
- $\texttt{Greater}(x, y)$ which is true if and only if $x > y$.

(a) Translate this sentence into English. $\exists x[\texttt{Prime}(x) \wedge \texttt{Divides}(3, x)]$ [3 points] **Solution:**

There is a positive prime integer that is divisible by 3.

(b) Translate this sentence into predicate logic: [3 points]

For every prime number x there is a unique positive integer y such that $y > 1$ and $y|x$. **Solution:**

Solution 1: $\forall x[\texttt{Prime}(x) \rightarrow \exists y(\texttt{Greater}(y, 1) \wedge \texttt{Divides}(y, x) \wedge \forall z(\texttt{Greater}(z, 1) \wedge \texttt{Divides}(z, x) \rightarrow z = y))]$

Solution 2: $\forall x[\texttt{Prime}(x) \rightarrow \exists y(\texttt{Greater}(y, 1) \wedge \texttt{Divides}(y, x) \wedge \forall z(z \neq y \rightarrow \neg\texttt{Greater}(z, 1) \vee \neg\texttt{Divides}(z, x)))]$

(c) Take the contrapositive of the implication in this sentence. Your final answer must be in English. You do not need to show work for this problem. [3 points]

If x, y are prime then $x \cdot y$ is not prime. **Solution:**

If $x \cdot y$ is prime, then x is not prime, or y is not prime.

(d) Negate the sentence from part (c). Give your answer in English. [3 points] **Solution:**

There exists positive integers x, y such that x is prime, y is prime, and $x \cdot y$ is prime.

Fill in the correct bubble [1 point each]

(e) The original statement in (c) is
◯ True
◯ False
Solution:

True.

(f) The contrapositive of the statement from (c) has
◯ The same truth value as the original statement from (c)
◯ The opposite truth value as the original statement from (c).
Solution:

The same truth value.

2. First Proof [12 points]

(a) Prove $(A \cup B) \setminus (A \cap B) \subseteq [A \setminus (A \cap B)] \cup [B \setminus (A \cap B)]$
You must format your proof as an English proof and structure your proof by introducing arbitrary element(s) of sets as appropriate.
We recommend drawing a picture of the sets for yourself so you see why the statement is true. [7 points]
Solution:

Let x be an arbitrary element of $A \cup B \setminus (A \cap B)$. By definition of $\setminus$, $x \in A \cup B$ and $x \notin A \cap B$. Since $x \in A \cup B$ (by definition of union) we have $x \in A$ or $x \in B$. We divide into cases:

Case 1: $x \in A$: In this case, we have $x \in A$ and $x \notin A \cap B$. This is exactly the definition of $x \in A \setminus (A \cap B)$. We therefore have $x \in A \setminus (A \cap B)$ or $x \in B \setminus (A \cap B)$. Then by definition of union, $x \in [A \setminus (A \cap B)] \cup [B \setminus (A \cap B)]$.

Case 2: $x \in B$: In this case, we have $x \in B$ and $x \notin A \cap B$. This is exactly the definition of $x \in B \setminus (A \cap B)$. We therefore have $x \in A \setminus (A \cap B)$ or $x \in B \setminus (A \cap B)$. Then by definition of union, $x \in [A \setminus (A \cap B)] \cup [B \setminus (A \cap B)]$.

Since x was arbitrary, $(A \cup B) \setminus (A \cap B) \subseteq [A \setminus (A \cap B)] \cup [B \setminus (A \cap B)]$.

(b) From (only) what you've written above, can you conclude: $(A \cup B) \setminus (A \cap B) = [A \setminus (A \cap B)] \cup [B \setminus (A \cap B)]$ (we've changed the subset from (a) to an equals sign).

If you can conclude the new statement, briefly (1-2 sentences) explain why.
If you cannot conclude the new statement, what statement do you still need to prove to get the new conclusion? (you can give you answer in English, set notation, or predicate logic notation – whichever you find most convenient). [2 points]

Solution:

You cannot conclude the new statement. You still need to prove the other direction of the inclusion, i.e. that $(A \cup B) \setminus (A \cap B) \supseteq [A \setminus (A \cap B)] \cup [B \setminus (A \cap B)]$

(c) Disprove the following statement: $(A \cup B) \setminus (A \cap B) = [(A \cup B) \setminus A] \cap [(A \cup B) \setminus B]$ [3 points **Solution:**

Consider $A = \{1, 2, 3\}$, $B = \{3, 4\}$

Then $(A \cup B) \setminus (A \cap B) = \{1, 2, 3, 4\} \setminus \{3\} = \{1, 2, 4\}$

While $[(A \cup B) \setminus A] \cap [(A \cup B) \setminus B] = [\{1, 2, 3, 4\} \setminus \{1, 2, 3\}] \cap [\{1, 2, 3, 4\} \setminus \{3, 4\}] = [\{4\}] \cap \{1, 2\} = \varnothing$.

The sets are not equal, as required.

Remark: Just about any sets A, B that are not comparable with subset (i.e., such that both $A \setminus B$ and $B \setminus A$ are nonempty) will produce a counter-example here.

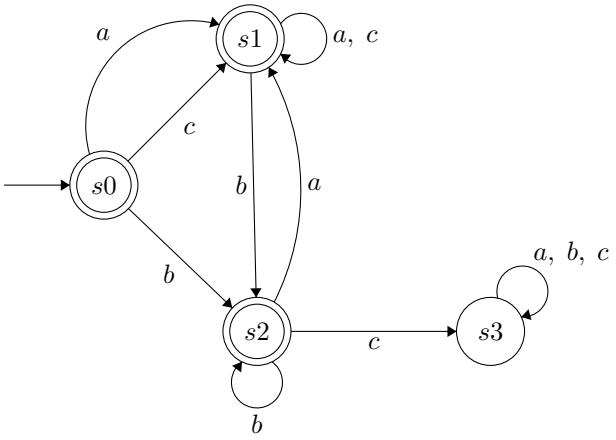
3. Models of Computation [11 points]

Let $\Sigma = \{a, b, c\}$ and let $L = \{w \in \Sigma^* : w$ does not contain bc as a substring$\}$.
In English, L is the language containing all strings over $\{a, b, c\}$ that do not have a substring bc.

(a) Write a regular expression that matches L. (No explanation required). [4 points] **Solution:**

There are many solutions. One is $c^*(b \cup ac^*)^*$.

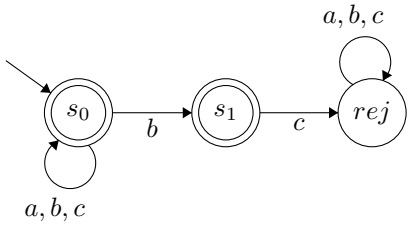
(b) Draw a DFA that accepts L. Include a brief description of how your machine works (You'll probably want about 2-3 sentences total, or a few words for each state about what it does). [4 points] **Solution:**



s_3 is a "garbage state" (we have seen bc and reject). The other states remember the most recent character, as long as we haven't seen bc yet (s_2 for just saw a b, s_1 for just saw an a or c, s_0 for no characters yet). The key idea is that as long as the last character is a b (we are in s_2 if we see a c we should reject. Until we see that substring we expect to accept (So s_0, s_1, s_2 are accepting states)

Remark: You could merge s_0 and s_1, and have a 3-state machine.

Your friend designs the **N**FA below, intending for it to accept L. They say that the machine, on seeing a b part of a bc will (magicially) transition right, and then on a c will transition right again to ensure if bc is a substring the machine rejects.



(c) There is at least one string that your friend's machine accepts that is not in L. [1.5 points]
Solution:

True. bc will be accepted (the machine will choose to stay in s_0).

(d) There is at least one string that your friend's machine **does not** accept that is in L. [1.5 points]
Solution:

False. This machine actually accepts Σ^*, as the machine can just stay in s_0.

4. *****Induction***** [20 points]

You're trying to write code to produce a string made up of $*$'s. At your disposal, you have three methods, described below:

```
/* This method prints 5 asterisks */
void print5() {
    print("*****");
}

/* This method prints 6 asterisks */
void print6() {
    print("******");
}

/* This method prints 13 asterisks */
void print13() {
    print("*************");
}
```

Show that, for every integer $n \geq 15$, we can use some combination of the methods above to print a string with n asterisks.

You must use strong or weak induction for this proof. We **strongly** recommend strong induction.
Remember to define a predicate $P(n)$.

Solution:

Let $P(n)$ be "the string with n asterisks can be printed". We show $P(n)$ holds for all $n \in \mathbb{Z}, n \geq 15$ by strong induction.

Base case ($n = 15$): `print5(); print5(); print5();` would print 15 stars. So $P(15)$ holds.

Base case ($n = 16$): `print5(); print5(); print6();` would print 16 stars. So $P(16)$ holds.

Base case ($n = 17$): `print5(); print6(); print6();` would print 17 stars. So $P(17)$ holds.

Base case ($n = 18$): `print6(); print6(); print6();` would print 18 stars. So $P(18)$ holds.

Base case ($n = 19$): `print13(); print6();` would print 19 stars. So $P(19)$ holds.

Inductive hypothesis: Suppose $P(15) \wedge P(16) \wedge P(17) \wedge P(18) \wedge P(19) \wedge \cdots \wedge P(k)$ for an arbitrary $k \in \mathbb{Z}, k \geq 19$.

Inductive step: We want to show $P(k+1)$, i.e. that $k+1$ asterisks can be printed.

By the inductive hypothesis, we can print $k-4$ asterisks (noting that $k \geq 19$, so $k-4 \geq 15$). Then, we can print an additional 5 asterisks by calling `print5()`. Thus, we can print $k-4+5 = k+1$ asterisks. So $P(k+1)$ holds.

By the principle of strong induction, $P(n)$ holds for all $n \in \mathbb{Z}, n \geq 15$.

5. Short Answer [12 points]

(a) Suppose you know $a \equiv b \pmod{10}$. Can you conclude that $a \equiv b \pmod{5}$? If so, briefly justify (not a full proof, just 1-3 sentences); if not give a counter-example. **Solution:**

Yes.

(Proof-ier) If $a \equiv b \pmod{10}$, then $10 \mid (a-b)$, so $a-b = 10k$ for some $k \in \mathbb{Z}$. In particular, $a-b = 5 \cdot (2k)$, where $2k \in \mathbb{Z}$, so $a \equiv b \pmod{5}$.

(Handwavier) If a and b are congruent mod 10, then they differ by a multiple of 10, which is "clearly" a multiple of 5, so they must be congruent mod 5.

(b) Suppose you know $a \equiv b \pmod{10}$. Can you conclude that $a \equiv b \pmod{20}$? If so, briefly justify (not a full proof, just 1-3 sentences); if not give a counter-example. **Solution:**

No.

Consider $a = 1$ and $b = 11$. Then $a - b = -1 \cdot 10$ where $-1 \in \mathbb{Z}$, so $a \equiv b \pmod{10}$, but $20 \nmid -10$, so $a \not\equiv b \pmod{20}$.

(c) Your goal is to show "if there is a context free grammar for L, then there is a DFA for L." using proof by contradiction.

Finish **just** the first sentence of the proof:

Proof. Suppose, for the sake of contradiction, there is a language L, such that □

Solution:

Suppose, for the sake of contradiction, there is a language L such that there is a context-free grammar for L, but no DFA for L exists.

(d) Let

$$f(n) = \begin{cases} f(n-1) + 2f(n-2) & \text{if } n > 2 \\ 2 & \text{if } n = 2 \\ 1 & \text{if } n = 1 \end{cases}$$

Imagine you define $P(n)$ to be "$2^{n-1} + 2 \cdot 2^{n-2}$." with the intent of proving $P(n)$ by induction. Briefly justify (1-2 sentences) why this is not a proper definition of $P(n)$.
Solution:

$\mathsf{P}(n)$ is not a valid predicate, since it evaluates to an integer, not a boolean.

6. II noitcudnI [20 points]

In this problem we define **CharTree**s as follows:

Basis Step: `Null` is a **CharTree**.
Recursive Step: If L, R are **CharTree**s and $c \in \Sigma$, then $\texttt{CharTree}(L, c, R)$ is also a **CharTree**.

Intuitively, a **CharTree** is a tree where the non-null nodes store a `char` data element.

We also define the following operations on **CharTree**s:

- The preorder function returns the preorder traversal of all elements in a **CharTree**.

$$\begin{aligned} \mathsf{preorder}(\texttt{Null}) &= \varepsilon \\ \mathsf{preorder}(\texttt{CharTree}(L, c, R)) &= c \cdot \mathsf{preorder}(L) \cdot \mathsf{preorder}(R) \end{aligned}$$

- The postorder function returns the postorder traversal of all elements in a **CharTree**.

$$\begin{aligned} \mathsf{postorder}(\texttt{Null}) &= \varepsilon \\ \mathsf{postorder}(\texttt{CharTree}(L, c, R)) &= \mathsf{postorder}(L) \cdot \mathsf{postorder}(R) \cdot c \end{aligned}$$

- The mirror function produces the mirror image of a **CharTree**.

$$\begin{aligned} \mathsf{mirror}(\texttt{Null}) &= \texttt{Null} \\ \mathsf{mirror}(\texttt{CharTree}(L, c, R)) &= \texttt{CharTree}(\mathsf{mirror}(R), c, \mathsf{mirror}(L)) \end{aligned}$$

Finally, for all strings x, let the "reversal" of x (in symbols x^R) produce the string in reverse order.

We have an example of all of these operations on the next page.

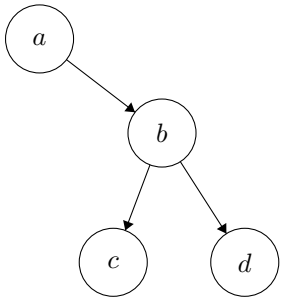
Now, let's actually get to the problem. Show, via structural induction, that for every **CharTree** T, the reversal of the preorder traversal of T is the same as the postorder traversal of the mirror of T.

In notation, you are to prove for every **CharTree**, T: $[\mathsf{preorder}(T)]^R = \mathsf{postorder}(\mathsf{mirror}(T))$.

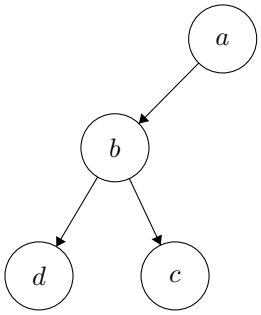
You may use the following facts without proving them:
Fact 1: For all strings $x_1, x_2, \ldots, x_k$: $(x_1 \cdot x_2 \cdot \cdots \cdot x_k)^R = x_k^R \cdot x_{k-1}^R \cdot \cdots \cdot x_1^R$
Fact 2: For every character c, $c^R = c$.
When applying these facts, you may treat characters as strings (and length-one strings as characters).

In your proof, apply at most one of the definitions/facts above in every step.

For more intuition on the rules, we have included an example below. **There are no new questions on this page.** But you may use the space below if you run out of room on the previous page.



Let T_i be the tree above.
$\mathsf{preorder}(T_i)$ ="abcd".
T_i is built as $(\texttt{null}, a, U)$
Where U is (V, b, W),
$V = (\texttt{null}, c, \texttt{null}), W = (\texttt{null}, d, \texttt{null})$.



This tree is $\mathsf{mirror}(T_i)$.
$\mathsf{postorder}(\mathsf{mirror}(T_i))$ ="dcba",
"dcba" is the reversal of "abcd" so
$[\mathsf{preorder}(T_i)]^R = \mathsf{postorder}(\mathsf{mirror}(T_i))$ holds for T_i

Solution:

Let $P(T)$ be "$[\mathsf{preorder}(T)]^R = \mathsf{postorder}(\mathsf{mirror}(T))$". We show $P(T)$ holds for all **CharTree**s T by structural induction.

Base case ($T = \texttt{Null}$): $\mathsf{preorder}(T)^R = \varepsilon^R = \varepsilon = \mathsf{postorder}(\texttt{Null}) = \mathsf{postorder}(\mathsf{mirror}(\texttt{Null}))$, so $P(\texttt{Null})$ holds.

Let T be an arbitrary T not covered by the base case. By the exclusion rule, $T = \texttt{CharTree}(L, c, R)$ for some character c and **CharTree**s L, R.

Inductive hypothesis: Suppose $P(L) \wedge P(R)$

Inductive step: We want to show $P(\texttt{CharTree}(L, c, R))$,
i.e. $[\mathsf{preorder}(\texttt{CharTree}(L, c, R))]^R = \mathsf{postorder}(\mathsf{mirror}(\texttt{CharTree}(L, c, R)))$.

$$
\begin{aligned}
\mathsf{preorder}(T)^R &= [c \cdot \mathsf{preorder}(L) \cdot \mathsf{preorder}(R)]^R && \text{defn of } \mathsf{preorder} \\
&= \mathsf{preorder}(R)^R \cdot \mathsf{preorder}(L)^R \cdot c^R && \text{Fact 1} \\
&= \mathsf{preorder}(R)^R \cdot \mathsf{preorder}(L)^R \cdot c && \text{Fact 2} \\
&= \mathsf{postorder}(\mathsf{mirror}(R)) \cdot \mathsf{postorder}(\mathsf{mirror}(L)) \cdot c && \text{by I.H.} \\
&= \mathsf{postorder}(\mathsf{CharTree}(\mathsf{mirror}(R), c, \mathsf{mirror}(L)) && \text{recursive defn of postorder} \\
&= \mathsf{postorder}(\mathsf{mirror}(\mathsf{CharTree}(L, c, R))) && \text{recursive defn of mirror} \\
&= \mathsf{postorder}(\mathsf{mirror}(T)) && \text{defn of } T
\end{aligned}
$$

So $P(\texttt{CharTree}(L, c, R))$ holds.

By the principle of induction, $P(T)$ holds for all **CharTree**s T.

7. Wait that's illegal [15 points]

Choose to do exactly one of these two problems.

(a) Let $L = \{a^{311}b^n c^{n+311} : n \geq 0\}$.

Using the proof technique from class, show that L is irregular. **Solution:**

Let $L = \{a^{311}b^n c^{n+311} : n \geq 0\}$. Suppose for contradiction that a DFA D accepts L.

Consider $S = \{a^{311}b^n c^{311} : n \geq 0\}$. Since S contains infinitely many strings and D has a finite number of states, two strings in S must end up in the same state.

Say these strings are $x = a^{311}b^i c^{311}$ and $y = a^{311}b^j c^{311}$ for some $i, j \geq 0$ such that $i \neq j$.

Append the string $z = c^i$ to both of these strings. The two resulting strings are:

$xz = a^{311}b^i c^{311+i}$ Note that $xz \in L$.

$yz = a^{311}b^i c^{311+j}$ Note that $yz \notin L$, since $i \neq j$.

Since xz and yz end up in the same state, but $xz \in L$ and $yz \notin L$, that state must be both an accept and reject state, which is a contradiction. Thus there is no DFA that recognizes L, so L is not regular.

(b) Let $\mathcal{F}$ be the set of all functions that take a binary string as input and produce either 'a' or 'b' as output.

In notation, $\mathcal{F} = \{f | f : \{0,1\}^* \to \{a, b\}\}$

Using the proof technique from class, show that $\mathcal{F}$ is uncountable.

Solution:

Suppose for the sake of contradiction that $\mathcal{F}$ is countable. Then there exists a listing of functions of $\mathcal{F}$ $f_1, f_2, \ldots$. We construct a new function $f' : \{0,1\}^* \to \{a, b\}$ as follows:

- $f'(0^n) = \begin{cases} b & \text{if } f_n(0^n) = a \\ a & \text{if } f_n(0^n) = b \end{cases}$
- $f'(s) = a$ for all other strings $s \in \{0,1\}^*$

Notice that f' differs from f_n on the input 0^n by construction for each $n \in \mathbb{N}$, so f' cannot be in the (infinite) list of functions, which contradicts our assumption that such a list exists. Therefore, $\mathcal{F}$ is uncountable.

Remark: There are other ways to construct the function f'. Another (more similar to class) would be to list binary strings in lexicographic order $(\varepsilon, 0, 1, 00, 01, ...)$ and have the k^{th} string in that order cause you to look at row k (i.e. the function $f_k()$).