CS 3120 Quizzes 1-5 (including 1-4 retakes)

This packet contains the quizzes for each module, to be taken during the final quiz day. This **cover sheet** is here to provide instructions, and to cover the questions until the quiz begins. **do not remove this cover sheet** until your proctor instructs you to do so.

You will have the entire class period to complete these quizzes. Each module quiz is two pages (front and back of one sheet of paper) worth of questions. Make sure to **write your name and computing id at the top of each individual quiz**. The quizzes are ordered in ascending order.

When you are done, you will come to the front of the room and cut off the staple to this quiz booklet. Afterward, you will discard this cover sheet and submit each quiz separately in a different pile. The proctors will be available at the front of the room to clarify this if you have any questions.

This quiz is CLOSED text book, closed-notes, closed-calculator, closed-cell phone, closed-computer, closed-neighbor, etc. Questions are worth different amounts, so be sure to look over all the questions and plan your time accordingly. Please sign the honor pledge below.

In theory, there is no difference between theory and practice.
But, in practice, there is.

THIS COVER SHEET WILL NOT BE SUBMITTED. DO NOT PUT WORK YOU WANT GRADED ON THIS PAGE

Quiz - Module 1: Proofs, Computers, and Cardinality

Name ______________________________

1. [8 points] Answer the following True/False questions.

A *Computer* is simply any mechanism that transforms *strings* into other *strings*	**True**	**False**
All *computers* must have access to *memory*	**True**	**False**
There are an infinite number of unique *Python programs*, but they do not cover / implement all possible functions	**True**	**False**
$\|\{0,1\}^\infty\| > \|\mathbb{N}\|$	**True**	**False**
If $\|S\| < \|\mathbb{N}\|$, then $\|S\|$ is countable	**True**	**False**
$f(n \in \mathbb{N}) = n + 1$ is a surjective function	**True**	**False**
Every subset of $\|\mathbb{N}\|$ is countable	**True**	**False**
The *diagonalization* proofs we have seen are examples of *proof by contradiction*	**True**	**False**

2. [6 points] For each claim and proof below, select whether the proof is valid or whether it has an error. For each item, the claim is true, so there is always some valid proof. *These are 2 points each*

$\forall_{n>2, n\%2=0}$, there exists a 3-regular (each node has degree 3) graph with n nodes	Let G have n nodes 0 to $n-1$. Add the edges $\{i, i+1\} \| 0 \le i \le n-2$, the edge $\{n-1, 0\}$ and the edges $\{i, i+\frac{n}{2} \| 0 \le i \le \frac{n}{2} - 1\}$	**Valid**	**Invalid**
There are 2^n binary strings of length n	for length 1, there are $2^1 = 2$ strings (0 and 1). Suppose there are 2^k strings of length k, to each of those we can add a 0 or a 1 to increase length by 1. Thus there are $2^k + 2^k = 2(2^k) = 2^{k+1}$ strings of length $k+1$	**Valid**	**Invalid**
For $n \in \mathbb{N}$, if $3n+2$ is even, then n is even	Suppose $3n+2$ is even, but n is odd. Since $3n+2$ is even, so is $3n$. $3n - n = 2n$, but an even number minus an odd number cannot equal an even number (contradiction). Thus, n is even.	**Valid**	**Invalid**

For this question, you will think about a *bijection* between $\{0,1\}^*$ and $\mathbb{N}$. First, you will look at a bijection that does NOT work and find one issue with it. Then, you will fix the bijection to make it correct. *Note: this is* $\{0,1\}^*$*, not* $\{0,1\}^\infty$.

3. [3 points] Consider the following potential bijection: $f(n \in \{0,1\}^*) = |n| + b(n)$, where $b(n)$ is the decimal integer value of binary number n (with one special case where b on the empty string returns 0). State whether this formula is not injective or whether it is not surjective, then provide specific inputs / outputs to prove your case.

4. [3 points] Now, write out an adjusted formula that fixes the issue, producing a valid *bijection*. You should only have to add a couple of characters to the formula. *Hint: Think about mapping onto the binary tree representation of binary numbers from the slides.*

NOTHING BELOW THIS POINT WILL BE GRADED

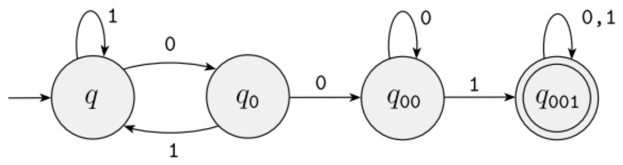
Quiz - Module 2: Regular Languages

Name ______________________________

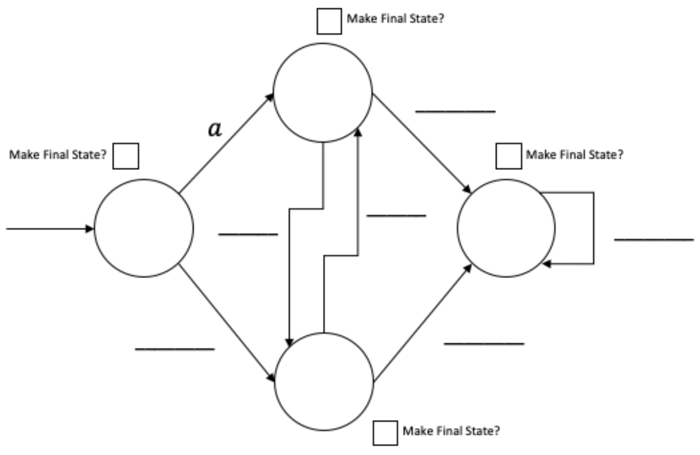
1. [8 points] Answer the following True/False questions.

$a^n b^n$ is not regular	**True**	**False**
Let n be the number of states in a *DFA*. If the *DFA* reads in $n-1$ characters of input, then at least one state was visited at least twice	**True**	**False**
A *DFA* is gauranteed to transition to a *new (different) state* after reading each input symbol	**True**	**False**
A *DFA* has greater recognizing power than an *NFA*	**True**	**False**
An *NFA* has greater recognizing power than a *DFA*	**True**	**False**
$a^*(a \cup b)^*$ is regular	**True**	**False**
Regular languages are closed under *Union*	**True**	**False**
Regular languages are closed under *reversal* (e.g., if 001 is in the original language then 100 is in the reversed language)	**True**	**False**

2. [2 points] Give a succinct regular expression for the dfa below.



3. [4 points] Draw a DFA with 4 states for the following language: $\{w \in \{a,b\}^* | \text{w has at least one repeated character}\}$. The states and transitions are drawn for you. You need to do two things: Label each transition arrow (do NOT add any more) and check the box next to states that should be final states. *You can add multiple characters to one transition blank (e.g., "a,b") if you need to.*



4. [6 points] On this page, you will look at some languages and mark whether or not you think they are regular or not. For each language below, select whether it is *regular* or whether it is *not regular*. *Hint: Some of these are tricky because they are written to look irregular but are regular! You should think about whether or not you can always pump strings in the language or not!*

$0^*1^* \cap 1^*0^*$	**Regular**	**Not Regular**
$\{1^k u \mid u \in \{0,1\}^*\}$ and u contains at LEAST k 1's for $k \geq 1$	**Regular**	**Not Regular**
$\{0^k 1 0^j \mid k \geq j\}$	**Regular**	**Not Regular**
$\{uww^R v \mid u, w, v \in \{0,1\}^*\}$ (Remember that w^R is the reverse of w)	**Regular**	**Not Regular**
$(0 \cup 1)^* \cup (0(1)^*0 \cup 1(0)^*1)^*$	**Regular**	**Not Regular**
w such that w contains the same number of 01 substrings as 10 substrings	**Regular**	**Not Regular**

NOTHING BELOW THIS POINT WILL BE GRADED

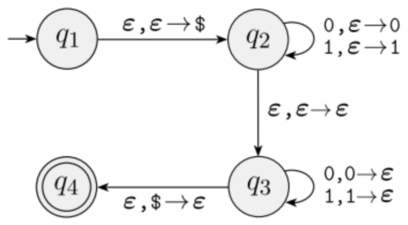
Quiz - Module 3: Context-Free Grammars

Name ______________________________

1. [8 points] Answer the following True/False questions.

Pushdown automata can recognize more functions than *DFA*s	**True**	**False**
If I give a *pushdown automata* a second stack, it can now recognize functions it could not before	**True**	**False**
A *pushdown automata* can recognize the language $a^n b^n c^n$	**True**	**False**
When converting a generic *PDA* into an equivalent grammar, the number of variables in the grammar was linear ($\Theta(n)$) relative to the number of states in the original *PDA*	**True**	**False**
Context-free grammars are closed under *intersection*	**True**	**False**
An *ambiguous grammar* is one in which one single string in the language has multiple unique derivations	**True**	**False**
A *PDA* has an infinite amount of stack space	**True**	**False**
When executing a *PDA*, the input is placed on the stack before execution begins	**True**	**False**

2. [3 points] Write an expression for the language recognized by the following *pushdown automata*. Please use the variable n if you need a variable.


q_1
$\varepsilon,\varepsilon\rightarrow\$$
q_2
$0,\varepsilon\rightarrow 0$
$1,\varepsilon\rightarrow 1$
$\varepsilon,\varepsilon\rightarrow\varepsilon$
q_3
$0,0\rightarrow\varepsilon$
$1,1\rightarrow\varepsilon$
q_4
$\varepsilon,\$\rightarrow\varepsilon$

3. [3 points] Consider the *CFG* for the language $a^x b^y c^y d^x$. Complete the grammar below. Note that S is the start variable.

$$S \rightarrow A$$

$$A \rightarrow a\_\_\_\_\_\_\_\_\_\_ \mid \_\_\_\_\_\_\_\_\_\_ \mid \epsilon$$

$$B \rightarrow \_\_\_\_\_\_\_\_\_\_ \mid \epsilon$$

On this page, you will answer some questions about the following homework problem: Let us define a new operation using the $\diamond$ symbol as such: if A and B are languages, then $A\diamond B = \{xy | x \in A, y \in B, |x| = |y|\}$. Prove that if A and B are regular languages, then $A\diamond B$ must be a context-free language.

4. [3 points] First, suppose we make a small change to the $\diamond$ operation: We remove the restriction that $|x| = |y|$. Is it still the case that $A\diamond B$ is context-free if A and B are regular, or is $A\diamond B$ simply regular? Explain your answer.

5. [3 points] Now, explain why $A\diamond B$ (with the length restriction) is context-free by describing how to use the *DFA*s for A and B to construct a *PDA* for $A\diamond B$.

NOTHING BELOW THIS POINT WILL BE GRADED

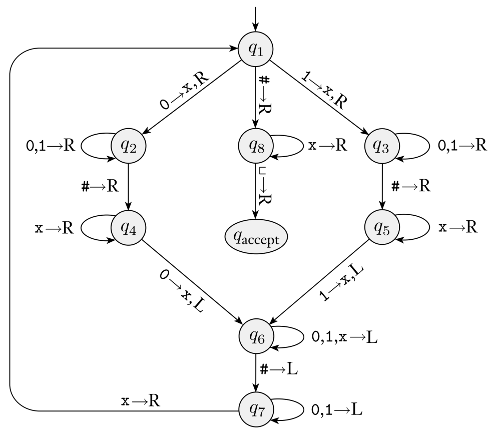
Quiz - Module 4: Turing Machines

Name ______________________________

1. [8 points] Answer the following True/False questions.

When simulating an *NTM* with a *DTM*, we updated the third tape after each simulation in order to execute a different branch in the *NTM*	**True**	**False**
The problem A_{TM}, as seen in class, is Turing-Recognizable	**True**	**False**
A *DTM* can loop forever, but an *NTM* will always halt because the branch of computation that accepts will always halt the machine	**True**	**False**
*NTM*s and *DTM*s decide / recognize the same set of functions because every *DTM* is an *NTM* and every *NTM* has an equivalent *DTM*	**True**	**False**
A *Turing Machine* with an infinite tape in both directions (left and right of the head's starting point) is equivalent in power to the *Turing Machine* we saw in class (tape is indexed from 0 with no negative indices).	**True**	**False**
A *Turing Machine* can simulate the behavior of another *Turing Machine*	**True**	**False**
A *Turing Machine* can be executed, given its own description as input	**True**	**False**
The *complement* of a language A can be harder to recognize than A	**True**	**False**

2. [4 points] Take a look at the Turing Machine below. For each of the strings to the right, will this machine accept, reject, or loop forever? Circle one answer for each string.



01#01	**Accept**	**Reject**	**Loop**
000#000	**Accept**	**Reject**	**Loop**
1101#1011	**Accept**	**Reject**	**Loop**
11#1	**Accept**	**Reject**	**Loop**

3. [2 points] In class, we saw that $\overline{A_{TM}}$ is not recognizable. Look at the prove below. The contains an error in exactly one bullet-point section. Identify the section with the error and circle it.

- Assume for sake of contradiction that $\overline{A_{TM}}$ is recognizable
- This means that $\overline{A_{TM}}$ (assumed) and A_{TM} (known) are both recognizable
- Thus, A_{TM} is decidable because when looking at its branches of computation, one path will always reject $\overline{A_{TM}}$ and accept A_{TM}
- Contradiction! A_{TM} is known to be undecidable.

On your homework, you proved that *Decidable Languages* are closed under *union*, *concatenation*, and *star*. For each of the proofs below, state whether it is correct or contains an error. If there is an error, write one or two sentences explaining the error and/or how to correct it.

4. [2 points] **Decidable Languages are closed under union**: consider decideable languages A and B, which each have a *DTM* M_A and M_B. You can recognize $A \cup B$ by simply introducing a new start state and using non-determinism to run M_A and M_B in parallel. Accept if either of the two original machines accepts.

5. [2 points] **Decidable Languages are closed under concatenation**: consider decideable languages A and B, which each have a *DTM* M_A and M_B. You can recognize A concatenate B by doing the following. For each possible dividing point in the input string $w = w_1w_2$, non-deterministcally guess that this is the correct dividing point for w. Simulate M_B on w_2 and M_A on w_1 and accept if both machines accept.

6. [2 points] **Decidable Languages are closed under star**: consider decideable language A which has a *DTM* M_A. You can recognize A^* by running M_A as normal, but epsilon transition back to the start state every time you enter a (previously final) state in M_A.

NOTHING BELOW THIS POINT WILL BE GRADED

Quiz - Module 5: Complexity Theory

Name ____________________

1. [8 points] Answer the following True/False questions.

P is the set of problems solvable in *polynomial time* by a *DTM*	**True**	**False**
NP is the set of problems that require (or seem to require) more than *polynomial time* to be solved by a *DTM*	**True**	**False**
If $P = NP$, then *NP-Complete* could be defined as the hardest problems in P	**True**	**False**
The *Cook-Levin Theorem* states that *SAT* is *NP-Complete*, which was proven via a reduction from *3-SAT*	**True**	**False**
3-SAT is easier to solve than *SAT*, because the structure of the formula makes it easy eliminate impossible settings for many variables	**True**	**False**
The runtime of an *NTM* is determined by the shortest branch of computation that leads to an accept state	**True**	**False**
Because I can verify *3-SAT* certificates in polynomial time, I can solve any *3-SAT* instance in exponential time	**True**	**False**
If a problem is solvable in *polynomial time* by an *NTM*, than the accepting branch of computation can be used as a *deterministic verifier* for that same problem	**True**	**False**

2. [6 points] Below, you will find several claims of reductions that exists between sets of problems. For every proposed reduction, select whether the reduction definitely exists, definitely does not exist, or we do not know. The first column asks you to consider an assumption when answering (if $P = NP$ or not).

Assumption	**Reduction**			
$P \neq NP$	SAT $\leq_p$ Vertex Cover	**Exists**	**Does Not Exist**	**Do Not Know**
$P \neq NP$	SAT $\leq_p$ Max-Flow	**Exists**	**Does Not Exist**	**Do Not Know**
$P \neq NP$	Max-Flow $\leq_p$ SAT	**Exists**	**Does Not Exist**	**Do Not Know**
$P = NP$	SAT $\leq_p$ Vertex Cover	**Exists**	**Does Not Exist**	**Do Not Know**
$P = NP$	SAT $\leq_p$ Max-Flow	**Exists**	**Does Not Exist**	**Do Not Know**
$P = NP$	Max-Flow $\leq_p$ SAT	**Exists**	**Does Not Exist**	**Do Not Know**

For this question, you will prove that a simple problem is *NP-Complete*. This is the *CIO Advertising* (*CIOA*) problem. You are president of a CIO called *DMT* and you want to advertise your club by sending gifts to other students. You have a list of students S that you could send a gift to. In addition, you have a list of other CIO clubs C that each student is a member of (C is a map with CIO club names as keys mapped to lists of students in those clubs). Given S and C, you want to send a gift to as few students as possible, such that at least one student member of each club in C receives a gift.

3. [2 points] Which of the following best describes how you would verify this problem in *polynomial time*? Circle the best answer.

   - Given the list of clubs and students, try each set of 1 student, then 2 students, etc. For each set, check if each club is covered.
   - Given students, clubs, and a subset of students to verify, loop through each club and ensure that one student from each club was selected.
   - Given students, clubs, and a subset of students to verify, loop through those students and make sure each is in at least one club.

4. [2 points] Now, we will reduce *3-SAT* to *CIOA*. Given a formula θ in *3-CNF* form, what would you map to your list of students S?

5. [2 points] To finish the reduction from the previous question, what would you map / how would you construct your list of clubs C (make sure to describe which students are in which clubs and why).

NOTHING BELOW THIS POINT WILL BE GRADED