

Final

Read this page and fill in your name, pledge, and email ID now.

Do not open past this page until instructed to do so.

Name: _____
UVA Email ID: _____

For this exam, you must **work alone**. You are not permitted to obtain help from people other than asking clarifying questions of the course staff. You are not permitted to provide help to others taking the exam. **You may not use any resources other than your brain and body, the one page of notes you prepared, and a simple writing implement like a pen or pencil.**

Sign below to indicate that you understand these expectations and can be trusted to behave honorably:

Signed: _____

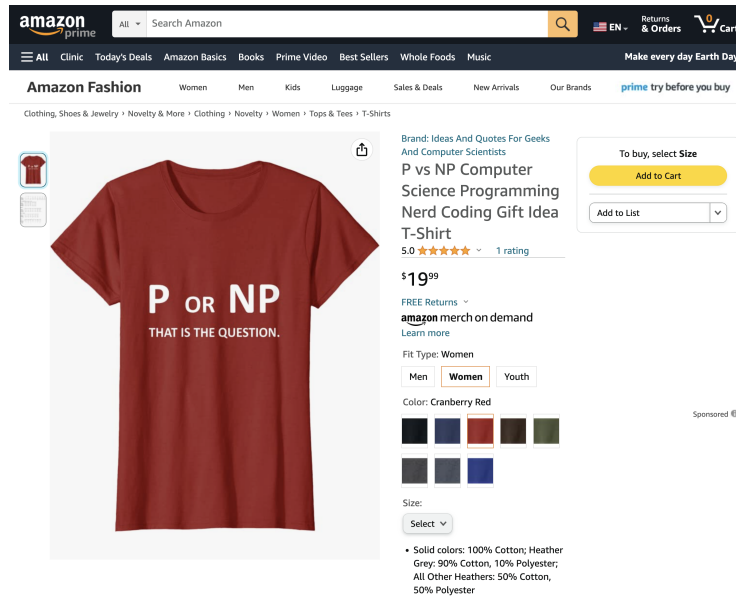
As discussed before, our goal is to design an exam that does not incentivize the intellectual dishonesty that is typically incentivized by school assignments and that you are all experts at, as demonstrated by your ability to achieve the level of success needed to be admitted to this selective University. Hence, please keep in mind that the exam will be graded in a way that is intended to *not reward intentionally obfuscated or deceptive answers* — if you do not know how to solve a problem, or get stuck at a step in a proof, it is much better to state that clearly and explain what you know that might be relevant or useful towards solving the problem, than to fabricate an answer that you know is wrong.

We will award fairly generous partial credit for answers that state that you do not know how to solve the asked problem, but either solve an easier one or show something you can do that is related to the given problem. Answers that we believe are deliberately deceptive might will not receive partial credit.

There are seven problems (some with multiple sub-parts) on this exam. Each of the questions awards 10 points for a “good as expected from an excellent cs3120 student” answer (you can also get up to **30 points for filling in the three blanks** on this cover page well enough so we can read your name and id). For each question, there is ample space provided to hold an excellent answer. If you need more space, you can use the backs of pages, but include clear markings and arrows to indicate the answer that should be graded. We will assume anything not inside an answer box or clearly marked from one, is your scratch work that should not be considered in scoring your answers.

Questions that Don't Belong on T-Shirts

1. Consider the following T-Shirt for sale at amazon.com:



(a) What is the answer to the “P OR NP” question posed by the T-shirt? (Mark the one best answer.)

- ☐ **True** (all languages are in P OR NP)
- ☐ \emptyset (the set P OR NP is empty)
- ☐ NP (P OR NP = NP)
- ☐ Unknown (the meaning of P OR NP depends on an open problem)

(b) Recall *LongestPath* is defined as:

Input: A graph, $G = (V, E)$; start and end nodes, $s, t \in V$; a path length, $k \in \mathbb{N}$

Output: 1 if there is a simple path from s to t in G with at least k hops. Otherwise, 0.

If someone finds an algorithm that solves *LongestPath* in $\Theta(n^{3120})$, which of the following would be true: (Check all statements that would be known to be true if such an algorithm were found)

- ☐ P OR NP = P
- ☐ $3SAT \in P$
- ☐ The encryption algorithms used to secure Amazon's web sessions might need to be redesigned.

Proving Uncomputability

2. In this question, your goal is to show that the function $IsXOR$ defined below is uncomputable.

Input: w , a description of a Turing machine.

Output: **1** if $\mathcal{M}(w)$ (i.e., the Turing machine described by w , as defined below) correctly computes the XOR function on two input bits b_1, b_0 ; namely, it eventually halts and outputs $XOR(b_0, b_1)$ when its input $x = b_0b_1$ has two bits. Otherwise, **0**.

That is, the input is a description of a machine which computes the logical XOR operation.

The (should be familiar to you) function \mathcal{M} is defined as:

$$\mathcal{M}(w) = \begin{cases} \text{TM described by } w & w \text{ corresponds to a valid Turing Machine} \\ \text{RejectMachine} & \text{Otherwise} \end{cases}$$

(a) Which of the strategies below could be employed to prove that $IsXOR$ is *uncomputable*? (Check the box for each method that could be used, even if they are more than one, no explanation is needed, but if you are unsure, you can provide one for potential partial credit.)

☐ Use Rice's theorem.

☐ Use Church-Turing Thesis.

☐ Reduce $HALTS$ to $IsXOR$.

☐ Reduce $LongestPath$ to $IsXOR$.

☐ Reduce $IsXOR$ to $HALTS$.

☐ Reduce $IsXOR$ to $LongestPath$.

(b) Use one of the strategies above to prove $IsXOR$ is uncomputable:

Countable, Uncountable, Unknown

3. For each set described below, indicate whether its cardinality is *finite*, *countably infinite* (countable and infinite), *uncountable*, or *unknown* (the answer depends on an open problem).

Circle one option and give a brief justification to support your answer.

(a) The set of all regular languages

Finite

Countably Infinite

Uncountable

Unknown

Justification:

(b) The set $\text{NP} \setminus \text{P}$ (the \setminus notation is the set subtraction operator, so this is the set of elements of NP that are not in P).

Finite

Countably Infinite

Uncountable

Unknown

Justification:

(c) The set S of all *infinite* binary strings that can be obtained by concatenating the binary representation of an *infinite sequence of* prime numbers. Namely, $S = \{x \mid x = x_1x_2\ldots \text{ such that } x_i \text{ is a prime number}\}$. For example, suppose p_i is the i th prime number. Then, the binary representation of the following strings are all in S : $p_1p_2p_3\ldots$, $p_1p_1p_1\ldots$ and $p_3p_1p_4p_1p_5\ldots$.

Finite

Countably Infinite

Uncountable

Unknown

Justification:

Always, Sometimes, Never

4. For any function $f : \{0, 1\}^* \rightarrow \{0, 1\}$, define $f_n : \{0, 1\}^n \rightarrow \{0, 1\}$ as the restricted version of f to only n -bit inputs. Suppose $s(n)$ is the minimum number of NAND gates that are needed to compute f_n where s is a function $\mathbb{N} \rightarrow \mathbb{N}$.

For each of the sub-question below, circle one to indicate whether the claim is *always true*, *possibly true*, or *never true* and provide a brief justification to support your answer (hint: a good justification for *possibly true* would be to show two choices for f , one where the answer is true and one where the answer is false; to justify an *always* or *never* answer, you need to show that the answer holds for *all* possible choices of f).

(a) f is computable

Always True

Possibly True

Never True

Justification:

(b) $\forall n \in \mathbb{N} : f_n$ is computable

Always True

Possibly True

Never True

Justification:

(c) $s(n) \leq 100 \times n \times 2^n$

Always True

Possibly True

Never True

Justification:

(d) f is NP-Complete.

Always True

Possibly True

Never True

Justification:

Complexity Classes

5. In each of the following sub-problems, you are given two complexity classes C_1 and C_2 where C_1 is a strict subset of C_2 (i.e., $C_1 \subsetneq C_2$). For each sub-part, describe a Boolean function (or equivalently a language) that is in complexity class C_2 but not in class C_1 . In all of them n is the length of the input. You do not have to provide a proof, but it should be clear why the language you have described is in C_2 but not in C_1 .

(a) C_1 = set of all functions that can be computed in $O(1)$ steps using a standard Turing machine; C_2 = set of all functions that can be computed in $O(n)$ steps using a standard Turing machine.

(b) C_1 = set of all regular languages, and $C_2 = \mathbf{P}$ is the set of all polynomial-time computable languages.

(c) C_1 = set of all computable languages, C_2 = set of all recognizable languages.

(d) C_1 = set of all recognizable languages, C_2 = set of all languages.

Regular Expressions and Automata

6. For each of the languages below, indicate if the language is *Regular* or *Not Regular*. Then, justify your answer. If the language is regular, provide either a regular expression or a finite automaton that corresponds to the language. If the language is not regular, provide an information explanation why.

(a) $L = \{x \mid x \in \{0, 1\}^*, \text{ and } x \text{ contains exactly three 0s}\}.$

Regular

Not Regular

(b) $L = \{x \mid x \in \{0, 1\}^* \text{ and } \exists y, z, w \in \{0, 1\}^* \text{ such that } x = y1z1w \text{ and } z \in \{0, 1\}^{2k+1} \text{ for some } k \in \mathbb{N} \text{ (that is, the length of } z \text{ is odd)}\}.$

Regular

Not Regular

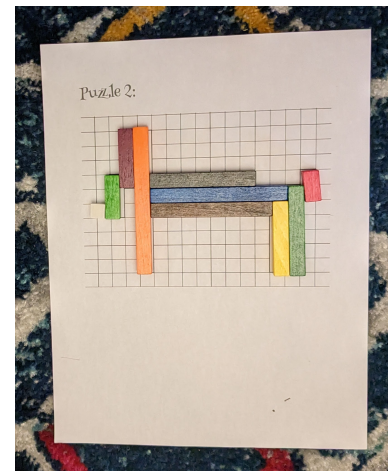
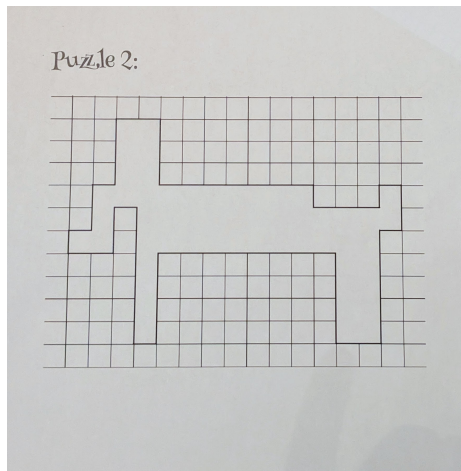
(c) $L = \{x \mid x = yz \text{ for } y, z \in \{0, 1\}^*, |y| = |z| \text{ and the number of 1s in } y \text{ is the same as the number of 1s in } z.\}$

Regular

Not Regular

Canine Creations

7. (Reminder: you may find the list of known NP-Complete problems on the provided definitions sheet helpful for some parts of this problem.) Cuisenaire Rods are puzzles where a finite set of integer-length rods are provided, along with a drawing of a shape, and the goal is to find a way to arrange the rods that exactly covers the shape.



We can model the game mathematically with the following Boolean function, *CuSAT*:

Input: S a finite multiset of elements from \mathbb{N} , which represents the available rods; and $P = \{((l_i, r_i), (m_i, t_i)) : l_i, r_i, m_i, t_i \in \mathbb{N}\}$, a representation of the shape to cover where each pair of coordinates describes a horizontal or vertical range of squares in a two-dimensional grid where the top-level square is $(0, 0)$.

Output: **1** if there is a way to cover the shape P using S (each rod can be used at most once, and can be rotated in any direction); **0** if there is no way to cover the shape P using S .

For the example in the pictures, $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and

$$P = \{((0, 6), (0, 6)), ((1, 4), (1, 6)), ((2, 1), (2, 4)), \dots, ((14, 4), (14, 5)),$$

and the arrangement in the rightmost picture shows that the output for this problem instance is **1**.

(a) Prove that *CuSAT* is in the class NP.

(b) Prove that $CuSAT$ is in the class NP-Complete.

End of Exam! (Optional Feedback questions on the last page)

Optional Feedback

(Optional) These questions are optional and will not adversely affect your grade.

Do you feel your performance on this exam will fairly reflect your understanding of the course material? If not, explain why.

Is there anything else you want us to know about your experience in this course, or things we could do in future courses to make it better? (Feel free to alternatively use this space to provide any other comments you want on the exam, the course so far, your thoughts on the CS curriculum, or just draw a picture or your favorite intractable problem.)