Practice Final Exam

Note: This is a practice final, based on exams used in previous semester. We have preserved the directions from the previous exam here, and encourage (but do not require) you to follow them. The practice exam is intended to help prepare you for the final, so we think it is useful to first try to take it yourself under exam-like conditions, and then to go back to the problems you had difficulty with to study them without time limits and with help from others.

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 in previous exams, our goal is to design exams that do 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 in high school to be admitted to the University. Hence, please keep in mind that the exam will be graded in a way that will 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.

Although fairly generous partial credit will be awarded for answers that state that you do not know how to solve the asked problem and 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 will receive negative scores (worse than that `0` that a blank answer receives for any question).

The exam has 12 questions, each of which awards a good answer with 10 points (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), so an exam with all good answers would be worth 150 points. For each question, we either identify the length an excellent answer should have, or else provide ample space 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.

Useful Definitions

Recall these definitions from class. We provide them here, and you can (carefully) rip this page out of the exam if it is helpful to you. These are all definitions you have seen before, and nothing should be surprising in them. We use $\mathbb{R}_+$ to denote the non-negative reals, and $\mathbb{R}^+$ to denote the positive reals (so $\mathbb{R}_+ = \mathbb{R}^+ \cup \{0\}$).

Definition 1 **(O)** A function $f : \mathbb{N} \rightarrow \mathbb{R}_+$ is in the set $O(g(n))$, defined for any function $g : \mathbb{N} \rightarrow \mathbb{R}_+$ iff there exist two constants $c \in \mathbb{R}^+, n_0 \in \mathbb{N}$ such that: $\forall n > n_0 . f(n) \leq cg(n)$.

Definition 2 **(Ω)** A function $f : \mathbb{N} \rightarrow \mathbb{R}_+$ is in the set $\Omega(g(n))$, defined for any function $g : \mathbb{N} \rightarrow \mathbb{R}_+$ iff there exist two constants $c \in \mathbb{R}^+, n_0 \in \mathbb{N}$ such that: $\forall n > n_0 . f(n) \geq cg(n)$.

Definition 3 **(Θ)** A function $f : \mathbb{N} \rightarrow \mathbb{R}_+$ is in the set $\Omega(g(n))$, defined for any function $g : \mathbb{N} \rightarrow \mathbb{R}_+$ iff $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$

Definition 4 (Busy Beaver Problem) For any $n \in \mathbb{N}$, define $BB_2(n)$ as the maximum number of steps for which a Turing Machine with n states and 2 symbols can execute and halt, starting from a blank tape.

Definition 5 (CNF) We say that a formula is in conjunctive normal form (CNF for short) if it is an AND of ORs of variables or their negations. E.g. $(x_7 \vee \overline{x_{22}} \vee x_{15}) \wedge (x_{37} \vee x_{22} \vee \overline{x_7})$ is in CNF. We say that it is k-CNF if there are exactly k variables per clause (group of variables combined with OR).

Definition 6 (DNF) We say that a formula is in disjunctive normal form (DNF for short) if it is an OR of ANDs of variables or their negations. E.g. $(x_7 \wedge \overline{x_{22}} \wedge x_{15}) \vee (x_{37} \wedge x_{22} \wedge \overline{x_7})$ is in DNF. We say that it is k-DNF if there are exactly k variables per clause (group of variables combined with AND).

True, False, or Unknown

1. For each of the following, circle one of the choices to indicate whether the statement is known to be *True*, is known to be *False*, or *Unknown* if its validity depends on something that is either currently unknown or not specified in the question.

Then, write a short justification to support your answer. When your answer is *Unknown*, your answer should make it clear what unknown the validity of the statement depends on (for example, that it is equivalent to a statement whose truth is currently unknown to anyone).

(a) The function, $XOR : \{0,1\}^* \rightarrow \{0,1\}$, which outputs the logical exclusive or of all the input bits, is in the complexity class P.

Circle one:

True *False* *Unknown*

Justification ($\leq$ 5 words):

(b) The function, XOR (from the previous question), is in the complexity class NP.

Circle one:

True *False* *Unknown*

Justification (3 symbols):

(c) The function, XOR (from the previous question), is in the complexity class $\mathsf{NP} - \mathsf{Complete}$.

Circle one:

True *False* *Unknown*

Justification ($\leq$ 15 words):

(d) If a function A in NP has an exponential lower bound (e.g., requires $\Omega(2^n)$ time to compute), then no function in NP-Complete can be computed in polynomial time.

Circle one:

True *False* *Unknown*

Justification ($\leq$ 3 sentences):

(e) If a function $Q : \{0,1\}^* \rightarrow \{0,1\}$ is computable, the function $\overline{Q}$ is computable where $\forall x \in \{0,1\}^* : \overline{Q}(x) = \mathsf{NOT}(Q(x))$.

Circle one:

True *False* *Unknown*

Justification ($\leq$ 3 sentences):

Proving Uncomputability

2. In this question, your goal is to show that the function $CELL_{15}$ defined below is uncomputable.

> **Input:** A string w that describes a Turing Machine.
>
> **Output: 1** if the machine described by w would write a 1 on the fifteenth cell on its tape when executed on a tape that is initially all blank. Otherwise, **0**.

That is, a machine which computes $CELL_{15}$ outputs **1** when the input describes a Turing Machine which, when run on a blank tape, at some points writes the symbol 1 to the tape cell at index 15 (counting from the start-of-tape symbol at index 0).

(a) Which strategy would show that $CELL_{15}$ is uncomputable? (Circle one, no explication needed.)

Use a machine that computes $CELL_{15}$ to compute $HALTS$.

Use a machine that computes $HALTS$ to compute$CELL_{15}$.

(b) Employ the strategy you chose in the previous question to show that $CELL_{15}$ is uncomputable.

Countable, Uncountable, Unknown

3. For each set described below, indicate whether its cardinality is *Countable*, *Uncountable*, or *Unknown* (not determined by the question if it is countable or uncountable). Circle one option and give a proof of your answer.

(a) The set of all grades that students will get on the final exam.

Countable *Uncountable* *Unknown*

Proof:

(b) The set of NAND circuits that compute XOR.

Countable *Uncountable* *Unknown*

Proof:

(c) The set of of all uncomputable languages.

Countable *Uncountable* *Unknown*

Proof:

Always, Sometimes, Never

4. For a function $f : \{0, 1\}^{3102} \to \{0, 1\}$ that can be implemented by a NAND circuit with s gates, which of the statements that follow would be *Always True, Possibly True* (meaning there are some functions f for which the statement is true and others for which is it false), or *Never True* (circle one option). Give a brief statement to justify your answer.

(a) f is computable

Always True *Possibly True* *Never True*

Justification ($\leq$ 5 words):

(b) $f \in \mathsf{NP}$

Always True *Possibly True* *Never True*

Justification ($\leq$ 5 words):

(c) f can be implemented using 3102 NAND gates.

Always True *Possibly True* *Never True*

Justification ($\leq$ 3 sentences):

Induction

5. Define the function $\texttt{ALT}_n : \{0,1\}^{2n} \to \{0,1\}$ such that for a string $w \in \{0,1\}^{2n}$ we say that $\texttt{ALT}_n(w) = 1$ provided $w \in (01)^*$. We could compute $\texttt{ALT}_1$ using the following straightline program:

```
def ALT1(x1,x2):
    diff = XOR(x1,x2)
    return AND(x2, diff)
```

We could then implement $\texttt{ALT}_n$ as follows:

```
def ALTn(x1,x2,...,x2n):
    diff = XOR(x1,x2)
    first = AND(x2,diff)
    rest = ALTn(x3,...,x2n)
    return AND(first, rest)
```

Suppose we have similarly implemented $\texttt{ALT}_{n-1}$, $\texttt{ALT}_{n-2}$, etc., (and all other dependent subroutines).

Show that the number of NAND gates needed to represent a circuit for $\texttt{ALT}_n$ is no more than $9n$ gates (hint: XOR requires 4 NAND gates and AND requires 3 NAND gates).

Complexity Classes

6. For an arbitrary given function A, for each of the complexity classes below, describe a way to prove that A belongs to the given class.

(a) P

(b) NP

(c) NP-Hard

(d) NP-Complete

(e) $O(n^2)$ (where n is the length of the input to A)

(f) $\Theta(1)$

(g) $\Omega(n)$ (where n is the length of the input to A)

7. Prove the following: If a function A in NP has an exponential lower bound (e.g. requires $\Omega(2^n)$ time to compute), then no language in NP-Complete can be computed in polynomial time.

Regular Expressions and Automata

8. For the following 4 sub-problems you will be asked to get both a regular expression and a finite state automaton for two different languages.

(a) Draw a finite state automaton (either an NFA or DFA) for the language:

$$\{x \in \{0,1\}^* \mid x \text{ as interpreted as a binary representation of a natural number is odd}\}$$

(note that the empty string is a binary representation of 0, which is even).

(b) Give a regular expression for the language:

$$\{x \in \{0,1\}^* \mid x \text{ as interpreted as a binary representation of a natural number is odd}\}$$

(note that the empty string is a binary representation of 0, which is even).

(c) Draw a finite state automaton (either an NFA or DFA) for the language: $\texttt{XOR} : \{0,1\}^* \rightarrow \{0,1\}$. In other words, the language $\{x \in \{0,1\}^* \mid \texttt{XOR}(x) = 1\}$.

(d) Give a regular expression for the language: $\texttt{XOR} : \{0,1\}^* \rightarrow \{0,1\}$. In other words, the language $\{x \in \{0,1\}^* \mid \texttt{XOR}(x) = 1\}$.

Models

9. List the essential things that are required to define a model of computing.

10. Describe how to show that two models of computing are equivalent.

11. A Turing Machine's configuration contains all the information needed to describe the current status of its computation (i.e., if I paused my computation then wrote the configuration down, I could resume the computation using what I had written). List all the necessary components of a Turing Machine's configuration.

Asymptotics

12. Let $f(n) = 8n^{4.5}$ and $g(n) = 5n^5$, which of the following are true? Support your answer to each part with a convincing argument.

(a) $f \in O(g)$

(b) $f \in \Omega(g)$

(c) $f \in \Theta(g)$