EECS 376 Final Exam

Instructions:
This exam is closed book, closed notebook. You may not provide your own scratch paper. No electronic devices are allowed. You may use **three** 8.5 × 11**-inch study sheets** (both sides) **that you prepared yourself**. Make sure you are taking the exam at the time slot and the classroom you were assigned by the staff. ***Please print your UNIQNAME at the top of each page.***

Any deviation from these rules may constitute an honor code violation. In addition, the staff reserves the right **not** to grade an exam taken in violation of this policy.

The exam consists of 10 multiple-choice questions, 3 short-answer questions, and 2 longer-answer questions. For the multiple-choice questions, please fill the selected circle completely and clearly. For the short- and long-answer sections, please write your answers clearly in the spaces provided. If you run out of room or need to start over, you may use the blank pages at the end, but you **MUST** make that clear in the space provided for the answer. The exam has 13 pages printed on both sides, including this page and the blank pages at the end.

You must leave all pages stapled together in their original order.

Write out and sign the full honor pledge below.

Honor pledge:
I have neither given nor received aid on this exam, nor have I concealed any violations of the Honor Code.
I will not discuss the exam with anyone before exam grades are released.
I attest that I am taking the exam at the time slot and the classroom I was assigned by the staff.

Signature: ______________________________

Name: ______________________________

Uniqname: ____________________

Standard Languages

You may rely on the following definitions without repeating them.

- $L_{\text{ACC}} = \{(\langle M \rangle, x) : M \text{ is a Turing machine that accepts } x\}$
- $L_{\text{HALT}} = \{(\langle M \rangle, x) : M \text{ is a Turing machine that halts on } x\}$
- $L_{\varepsilon\text{-HALT}} = \{\langle M \rangle : M \text{ is a Turing machine that halts on } \varepsilon\}$
- $L_{\emptyset} = \{\langle M \rangle : M \text{ is a Turing machine for which } L(M) = \emptyset\}$
- $L_{\text{EQ}} = \{(\langle M_1 \rangle, \langle M_2 \rangle) : M_1, M_2 \text{ are Turing machines for which } L(M_1) = L(M_2)\}$
- $\text{SAT} = \{\phi : \phi \text{ is a sastisfiable Boolean formula}\}$
- $\text{3SAT} = \{\phi : \phi \text{ is a satisfiable 3CNF formula}\}$
- $\textsc{Clique} = \{(G, k) : G \text{ is an undirected graph with a clique of size } k\}$
- $\textsc{VertexCover} = \{(G, k) : G \text{ is an undirected graph with a vertex cover of size } k\}$
- $\textsc{IndependentSet} = \{(G, k) : G \text{ is an undirected graph with an independent set of size } k\}$
- $\textsc{HamiltonianCycle} = \{G : G \text{ is a graph with a Hamiltonian cycle}\}$
- $\textsc{HamiltonianPath} = \{(G, s, t) : G \text{ is a graph with a Hamiltonian path from } s \text{ to } t\}$

UNIQNAME (print): ______________________________

Multiple Choice – 36 points (+4 extra credit)

1. Select the option that **correctly describes the following statement**:

   Some NP-Complete language is decidable in polynomial time if and only if *every* NP-Complete language is decidable in polynomial time.

   ◯ Known to be true
   ◯ Known to be false
   ◯ Unknown whether true or false

2. Select the option that **correctly describes the following statement**:

   The language

   $$\textsc{LogSAT} = \{\phi : \phi \text{ is a satisfiable Boolean formula having } 10\log_2|\phi| \text{ variables }\}$$

   is in P.

   (Recall that $|\phi|$ denotes the size of ϕ, when represented as a bit string.)

   ◯ Known to be true
   ◯ Known to be false
   ◯ Unknown whether true or false

3. Suppose that X is a non-negative random variable for which $\mathbb{E}[X] = 1/2$. Select the **tightest upper bound that *necessarily* holds** for $\Pr[X \geq 5/7]$.

   ◯ $e^{-(3/14)^2}$
   ◯ $7/10$
   ◯ $5/7$
   ◯ 1

4. Let $n = 33$ be a public RSA modulus and $e = 7$ be the public exponent. **Select the corresponding decryption exponent** d and the **decrypted message** m for the ciphertext $c = 2$.

   ◯ $d = 3$, $m = 29$
   ◯ $d = 3$, $m = 8$
   ◯ $d = 19$, $m = 17$
   ◯ $d = 19$, $m = 29$

5. In an alternate (more chaotic) universe, n EECS 376 students take the final exam, and each graded exam is returned to a student *chosen uniformly at random*—but these choices *may or may not be independent*. In particular, it might be the case that a student could receive zero, one, or more than one exam(s).

   Select the option that correctly describes the **expected number of students who receive their own exams**.

   ◯ It is $1/n$ if the choices are independent; otherwise, there is not enough information to determine the answer
   ◯ It is $1/n$ regardless of whether the choices are independent
   ◯ It is 1 if the choices are independent; otherwise, there is not enough information to determine the answer
   ◯ It is 1 regardless of whether the choices are independent
   ◯ Not enough information has been given to determine the answer, regardless of whether the choices are independent

6. Select the **one language-input pair** for which the Cook-Levin theorem gives a **reduction from the language** to SAT, and **outputs a satisfiable Boolean formula** given the input.
   - ◯ Language: L_{HALT}
     Input: $\langle M, x\rangle$ where M is a Turing machine and $M(x)$ rejects
   - ◯ Language: SAT
     Input: Boolean formula $\phi = (x \text{ AND NOT}(x)) \text{ OR } (y \text{ AND NOT}(y))$
   - ◯ Language: Clique
     Input: $(G, 203)$, where G is a graph that has a clique of size 376
   - ◯ Language: $\text{GCD} = \{(x, y, g) : g = \gcd(x, y)\}$
     Input: $(203, 376, 3)$

7. Select the **one statement that is known to be true, assuming that** $\mathsf{P} \neq \mathsf{NP}$.
   - ◯ No algorithm correctly solves all SAT instances having up to 1 billion variables in less than 1 second (on a standard laptop).
   - ◯ No polynomial-time algorithm correctly solves $\geq 99.9\%$ of all VertexCover instances of any given size.
   - ◯ There is no polynomial-time algorithm for any language in NP.
   - ◯ There is no polynomial-time algorithm for any NP-hard language.

8. Define the language $L_{203} = \{\langle M\rangle : M \text{ does } \textit{not} \text{ accept the string "203"}\}$, and consider the following Turing machine:

```
1: function R(⟨M⟩)
2:     Run M("203") and output the opposite of whatever it outputs.
```

   Select the **one true statement** about this machine.
   - ◯ It does not recognize L_{203} because it does not accept every $\langle M\rangle \in L_{203}$
   - ◯ It does not recognize L_{203} because it does not reject or loop on every $\langle M\rangle \notin L_{203}$
   - ◯ It recognizes L_{203}, but does not decide L_{203}
   - ◯ It decides L_{203}

9. **Assuming that** $\mathsf{P} = \mathsf{NP}$, select the option that **correctly describes the following statement**:
   There is a polynomial-time algorithm that, given only the public information from a run of the Diffie-Hellman protocol, outputs the resulting shared key.
   - ◯ Known to be true
   - ◯ Known to be false
   - ◯ Unknown whether true or false

10. **(Extra credit.)** What are the names of Prof. Chris's kittens?
    - ◯ Itchy and Scratchy
    - ◯ Biggie and Smalls
    - ◯ The Killers
    - ◯ Any/all of the above

UNIQNAME (print): ____________________

Short Answer – 24 points

1. Recall the *decision* version of the subset-sum problem:

$$\textsc{SubsetSum} = \big\{(A,t) : \exists\, S \subseteq A \text{ whose elements sum to } t \in \mathbb{N}\big\}.$$

In this problem, **suppose that there is an efficient algorithm D that decides SubsetSum**.

The following *incomplete* algorithm uses D to efficiently solve the subset-sum *search* problem: given input (A, t), it should output a subset $S \subseteq A$ whose elements sum to t, if such a subset exists.

```
function SubsetSumSearch($A = \{a_1, \ldots, a_n\}, t$)
    if TODO#1 then
        return “no such subset exists”
    $S \leftarrow \emptyset$, $t' \leftarrow t$
    for $i = n$ down to 1 (inclusive) do
        if $D(\{a_1, \ldots, a_{i-1}\}$, TODO#2) rejects then
            $S \leftarrow$ TODO#3
            $t' \leftarrow$ TODO#4
    return $S$
```

State what TODO#1 through TODO#4 should be to make the algorithm correct and efficient. You do *not* need to justify these answers.

UNIQNAME (print): ______________________________

2. After studying all week for the EECS 376 final exam, you have prepared 3 perfect pages of notes, which have 240, 560, and 480 symbols, respectively. You need to print your masterpiece, but the University's printers are malfunctioning! Specifically, *each symbol* in your notes is *independently* printed with probability 3/4, and not printed with probability 1/4.

   Thanks to your diligent studying, you will be able to understand a page as long as *more than half* of its symbols are printed. (The symbols do not need to be contiguous.)

   In the following, **give the tightest bounds you can** for the probabilities in question.

   (a) Use the Chernoff-Hoeffding bound (with brief justification) to give an **upper bound** for the probability that you **will *not* be able to understand your first page**, i.e., at most half of its symbols will be printed.

   Also **do the same for the second and third pages**.

   Simplify the expressions as much as possible, but you *do not* need to evaluate any exponentials.

   (b) Give, with brief justification, a **lower bound** for the probability that **you *will* be able to understand *all* of your pages**. (Simplify as in the previous part.)

UNIQNAME (print): ______________________________

3. Consider a load balancer that needs to assign several tasks to worker machines, without overloading any machine. Let $T_i \in [0, 4]$ be the amount of work the ith task requires. **No machine should get more than 4 units of work** in total. The goal is to **minimize the number of machines** to which tasks are assigned. Consider the following assignment algorithm:

```
1: function AssignTasks($T_1, \ldots, T_k$)
2:     for $i = 1$ to $k$ do
3:         if task $i$ can be assigned to some already-provisioned machine (without overloading it) then
4:             assign task $i$ to such a machine
5:         else
6:             provision a new machine and assign task $i$ to it
7:     return the assignment
```

(a) Suppose that AssignTasks($T_1, \ldots, T_k$) provisions M machines. Show that $\sum_{i=1}^{k} T_i > 2(M-1)$.
Hint: argue that at most one provisioned machine can end up being assigned ≤ 2 units of work.

(b) Let OPT be the minimum number of machines needed. Show that $M < 2 \cdot \mathsf{OPT} + 1$ (and hence $M \leq 2 \cdot \mathsf{OPT}$), where again M is the number of machines provisioned by AssignTasks($T_1, \ldots, T_k$). (You may use the statement from the previous part even if you did not manage to prove it.)
Hint: lower bound OPT in terms of $\sum_{i=1}^{k} T_i$.

4. (Note: This question was not on the original exam.)

   An undirected graph $G = (V, E)$ is *3-colorable* if it is possible to assign each vertex one of three colors named R,W,B so that for every edge $(u, v) \in E$, the vertices u, v are have *different* colors.

   For example, a triangle graph is 3-colorable because we can assign each vertex a different color, but a complete graph on four vertices is not 3-colorable because any assignment of colors will result in (at least) two vertices having the same color, so the edge between those two vertices will be invalid. It is known that the decision problem of determining whether a given graph is 3-colorable is NP-complete.

   Consider the following zero-knowledge proof that a given graph $G = (V, E)$ is 3-colorable.

   - The prover (Merlin) starts with some fixed 3-coloring of G, and *recolors* it by randomly 'shuffling' the colors R,W,B. (For example, for the shuffling R,W,B $\to$ B,W,R, it would recolor every R vertex as B, every W vertex as W, and every B vertex as R.) It then conceals each vertex with a removable opaque cover, and shows the graph with its concealed vertices to the verifier (Arthur).
   - The verifier checks that the covered graph is indeed G (has the same vertices and edges), and then 'challenges' the prover by choosing a uniformly random edge $(u, v) \in E$.
   - The prover uncovers just the vertices u, v, revealing their assigned colors.
   - If u and v have *different* (valid) colors, the verifier accepts; otherwise, it rejects.

   This basic iteration can be repeated multiple times (reshuffling the colors each time) to improve the soundness, but below we consider only a single iteration.

   (a) Suppose that G is *not* 3-colorable, but the prover tries to 'fool' the verifier into accepting. Give, with brief justification, the tightest possible upper bound on the probability (over the verifier's random choice of challenge) that the prover can succeed.

   (b) Give a brief justification (1–2 sentences) why the protocol is zero knowledge.

UNIQNAME (print): ____________________

Longer Answer – 40 points

1. Define the language

$$\textsc{SetPacking} = \left\{ (A, S_1, \ldots, S_n, k) : \begin{array}{l} \text{each } S_i \subseteq A, \text{ and some } k \text{ of them} \\ \text{are pairwise disjoint} \end{array} \right\}.$$

(Recall that sets are pairwise disjoint if the intersection of any two of them is the empty set.)

Show that SetPacking is NP-Hard by proving that IndependentSet $\leq_p$ SetPacking.

Hint: give a reduction that outputs suitable subsets of *edges*.

UNIQNAME (print): ____________________

2. In this problem, we consider a simple randomized approximation algorithm for the maximum independent set problem. The input is an undirected graph G with n vertices, each of which has *at most d neighbors* (and no self-loops). The algorithm works as follows:

   - Choose a uniformly random permutation (ordering) of the vertices, and assign each vertex its position in the permutation, from $1, \ldots, n$.
   - Output the set of *peak* vertices, where a vertex is a peak if it is assigned a *larger* number than all its neighbors.

   For example, if G is a cycle consisting of vertices u, v, w (so $n = 3$ and $d = 2$), and the permutation is (v, w, u), then u, v, w are assigned $3, 1, 2$, respectively, and u is the only peak vertex.

   (a) Define suitable indicator random variables and show that their sum is equal to the number of peak vertices.

   (b) Prove that the expected number of peak vertices is at least $n/(d + 1)$.
   Hint: You may use (without proof) the fact that for any subset S of vertices, by symmetry, each vertex in S is equally likely to be assigned the largest number among all the vertices in S.

   (c) Prove that the set of peak vertices is an independent set of G, and that its expected size is at least $1/(d + 1)$ times the size of any (maximum) independent set of G.

UNIQNAME (print): ______________________________

This page intentionally left blank.

If you have answer(s) here, be sure to write “answer continues on page 11” in the appropriate box.

UNIQNAME (print): ______________________________

This page intentionally left blank.
If you have answer(s) here, be sure to write “answer continues on page 12” in the appropriate box.

UNIQNAME (print): ____________________

This page intentionally left blank.
If you have answer(s) here, be sure to write “answer continues on page 13” in the appropriate box.