**UNIQNAME** (print clearly): ______________________________



# EECS 376 Final Exam Solutions, Fall 2023

**Instructions:** This exam is closed book, closed notebook. No electronic devices are allowed. You may use two $8.5 \times 11$-inch study sheets. Make sure you are taking the exam at the time slot and the classroom you were assigned by the staff.

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

The exam consists of **5** multiple-choice questions, **3** short-answer questions, and **3** longer-answer questions. The multiple choice questions may have more than one correct answer; ***mark all correct answers***. 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 page at the end, but you **MUST** make that clear in the space provided for the answer. The exam has 9 pages printed on both sides, including this page and the blank page at the end.

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

*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: ______________________________

**PRINT YOUR NAME/UNIQNAME AS CLEARLY AS YOU POSSIBLY CAN:**

Full Name: ______________________________

Uniqname: ______________________________

## Questionnaire (NOT GRADED)

**Answers to these questions will have no impact on your exam grade or final grade.**

1. The percentage of lectures that I attended is roughly:

- ☐ 0-20%.
- ☐ 20-40%.
- ☐ 40-60%.
- ☐ 60-80%.
- ☐ 80-100%.

2. What class resources did you routinely use? (Mark all that apply.)

- ☐ Live Lectures
- ☐ Recorded Lectures
- ☐ Discussion Sections
- ☐ Piazza
- ☐ Office Hours
- ☐ Course Notes on eecs376.org

**UNIQNAME** (print clearly): ____________________

# Multiple Choice — 6 Points Each

**In all multiple choice questions, fill in <u>all</u> correct boxes and <u>no</u> incorrect boxes.**

1. Which of the following languages are *guaranteed* to be in NP?

   ■ $L_1 \cap L_2$, where $L_1 \in \mathsf{NP}$ and $L_2$ is NP-complete.
   ☐ $L$, where it is known that $L \notin P$.
   ☐ $L$, where $L \leq_T L_{\text{HALT}}$, where $L_{\text{HALT}} = \{\langle M, x\rangle \mid \text{TM } M \text{ halts on input } x\}$.
   ■ $L_1 \cup L_2$, where $L_1$ and $L_2$ are both in NP.
   ☐ $L$, where $\overline{L} \in \mathsf{NP}$.

2. Suppose language $A$ is NP-Complete and language $B = \{x \in \{0,1\}^* \mid x \text{ is a palindrome}\}$. Which of the following are true?

   ☐ If $B \leq_p A$ then $P = NP$.
   ■ If $A \leq_p B$ then $P = NP$.
   ■ $A \leq_T B$.
   ■ $B \leq_T A$.

3. Suppose an $O(m^2)$-time algorithm is discovered that, given a boolean 3CNF formula $\phi$ with $n$ variables and $m$ clauses, returns a satisfying assignment, if there is one. What are the *guaranteed* consequences of this? (MAX-CLIQUE is the problem: given $G = (V, E)$ to find the largest $U \subseteq V$ such that $U$ forms a clique in $G$.)

   ☐ There is an $O(N^2)$-time algorithm for *every* problem in NP, where $N$ is the length of the input.
   ■ $L = \{x \in \{0,1\}^* \mid \ x = 0^n 1^n \text{ for some } n \geq 0\}$ is NP-complete.
   ■ $\mathsf{P} = \mathsf{NP}$.
   ■ There is an efficient 99/100-approximation algorithm for MAX-CLIQUE.
   ■ The Diffie-Hellman protocol can be broken in polynomial time.

4. Suppose $A$ is a 1/2-approximation algorithm for the MAX-CLIQUE problem. Then

   ☐ $A(G)$ is guaranteed to return a clique in $G$ of size $n/2$ if such a clique exists, where $G = (V, E)$ and $|V| = n$.
   ☐ $A(G)$ is guaranteed to return a clique of at least twice the maximum clique in $G$.
   ■ $A(G)$ is guaranteed to return a clique of size at least half the largest clique in $G$.
   ☐ $A(G)$ *always* returns a larger clique than what a 1/4-approximation algorithm for MAX-CLIQUE would return.

5. Which of these decision problems is in the set NP?

   ■ Given a graph $G$, decide if it has a vertex cover of size $k$.
   ■ Given a set of distinct integers, decide if they can be partitioned into three sets with the same sum.
   ■ Given a prime $p$, a generator $g$, and $t \in \{1, \ldots, p-1\}$, decide if the integer $i \in \{1, \ldots, p-1\}$ with $g^i \equiv t \pmod{p}$ is even.
   ■ Given two integers $p, q$, decide whether their greatest common divisor is less than 10.
   ☐ Given a Turing machine $M$, decide if it halts when the input is the empty string.

# Short Answer — 11 Points Each

1. Alice and Bob agree on the prime $p = 17$ and generator $g = 6$, and wish to establish a shared secret key $k$ using the Diffie-Hellman protocol. Suppose Alice privately chooses $a = 3$ and Bob privately chooses $b = 12$. What is Alice's message to Bob? What is Bob's message to Alice? What is the secret key $k$?

**Solution: Write all calculations clearly for partial credit:**

First we compute some useful powers of 6, modulo 17.

$$\begin{aligned} 6^2 &\equiv 36 \equiv 2 \pmod{17} \\ 6^3 &\equiv 2 \cdot 6 \equiv 12 \pmod{17} \\ 6^4 &\equiv (6^2)^2 \equiv 2^2 \equiv 4 \pmod{17} \\ 6^{12} &\equiv (6^2)^6 \equiv 2^6 \equiv 64 \equiv 13 \pmod{17} \end{aligned}$$

Alice's public key: $6^3 \mod 17 = 12$.

Bob's public key: $6^{12} \mod 17 = 13$.

Shared secret key: Using Fermat's Little Theorem to simplify the computation,

$$6^{3 \cdot 12} \equiv 6^{36} \equiv 6^{16 \cdot 2 + 4} \equiv (6^{16})^2 \cdot 6^4 \equiv 4 \pmod{17}.$$

All three answers should be integers in $\{1, \ldots, 16\}$.

| Alice's message to Bob: | Bob's message to Alice: | The Shared Secret $k$ |
|---|---|---|
| 12 | 13 | 4 |

2. Alice uses modulus $n = 65$ and RSA public key $e = 29$. Determine a possible private key $d$ and calculate Alice's RSA-signature for the message $m = 3$.

**Solution: Write all calculations clearly for credit:**

We have $n = 65 = 5 \cdot 13$, so $\phi(n) = (5 - 1)(13 - 1) = 48$.
We know $de \equiv 1 \pmod{48}$. The first few multiples of 48 are $48, 96, 144$.
$145 = 29 \cdot 5$, so $d = 5$ is a possible private key.
$m^d \equiv 3^5 \equiv 3 \cdot 81 \equiv 3 \cdot 16 = 48 \pmod{65}$. The signature is 48.

| Private key $d$ (in $\{1, \ldots, 64\}$): | Signature for message $m = 3$ (in $\{1, \ldots, 64\}$): |
|---|---|
| 5 | 48 |

**UNIQNAME** (print clearly): ______________________________

3. You've invented a new, ultrafast randomized primality testing algorithm but it makes lots of mistakes, both false positives and false negatives. If $n$ is prime, it reports "PRIME" with probability 3/4 and "COMPOSITE" with probability 1/4. If $n$ is composite, it reports "COMPOSITE" with probability 3/4 and "PRIME" with probability 1/4. Explain how to reduce the error probability of this algorithm from 1/4 to any desired $\delta > 0$. How many times do you need to call the primality tester, as a function of $\delta$? (Hint: consider returning a majority vote of the answers.)

**Solution:**

We call the primality tester $k$ times can take the majority vote.

Let $X_1, \ldots, X_k$ be the error indicators, i.e., $X_i = 1$ if the $i$th call reports an incorrect answer and 0 otherwise. We have $\mathbf{E}[X_i] = 1/4$. Defining $X = \sum_{i=1}^{k} X_i$, we have $\mathbf{E}[X/k] = 1/4$ and will make an error if $X/k \geq 1/2$. Applying Chernoff-Hoeffding bounds with $\epsilon = 1/4$,

$$\Pr\left[\frac{X}{k} \geq \frac{1}{2}\right] \leq e^{-2(1/4)^2 k} = e^{-k/8}.$$

For this to be at most $\delta$, we can set $k = \lceil 8 \ln(1/\delta) \rceil$.

Number of calls to primality tester, in terms of $\delta$ : $\lceil 8 \ln(1/\delta) \rceil$

**UNIQNAME** (print clearly): ____________________

# Long Answer — 12 Points Each

1. Recall that a *tree* is a connected, acyclic graph, and that a *leaf* in a tree is a vertex with degree 1 (incident to 1 edge). The *minimum-leaf* problem is to find a spanning tree with the fewest number of leaves. We express this as a decision problem MIN-LEAF.

$$\text{MIN-LEAF} = \{\langle G, k\rangle \mid \text{ undirected graph } G \text{ contains a spanning tree with at most } k \text{ leaves}\}$$

Prove that MIN-LEAF is NP-hard by reducing HAM-PATH to MIN-LEAF.

$$\text{HAM-PATH} = \Big\{\langle G', s, t\rangle \mid \text{ undirected graph } G' = (V', E') \text{ contains a Hamiltonian path from } s \text{ to } t,\ s, t \in V'.\Big\}$$

**Solution:** The reduction function is as follows:

$f(G' = (V', E'), s, t)$
1. Create a graph $G = (V, E)$ where
   $V = V' \cup \{s_0, t_0\}$
   $E = E' \cup \{\{s_0, s\}, \{t_0, t\}\}$
2. Return $(G, 2)$ \\\\ i.e., $k = 2$

In other words, add two new vertices $s_0, t_0$, and attach them with two new edges to $s$ and $t$, respectively.

Suppose $(G', s, t) \in$ HAM-PATH. Let $P = (s, \ldots, t)$ be a Hamiltonian path from $s$ to $t$ in $G'$. Then $\{s_0, s\}, P, \{t, t_0\}$ is a spanning tree of $G$ with two leaves, namely $s_0, t_0$, so $(G, 2) \in$ MIN-LEAF.

Now suppose $f(G', s, t) = (G, 2) \in$ MIN-LEAF. Let $T$ be a spanning tree of $G$ with at most 2 leaves. Every such tree must consist of a single path. The endpoints of the path must be $s_0, t_0$ since they are each incident to only one edge in $G$, hence $T$ is the path $(s_0, s, \ldots, t, t_0)$. Trimming off $s_0, t_0$ from each end, we have a Hamiltonian path in $G'$, hence $(G', s, t) \in$ HAM-PATH.

2. In the EQUITABLE-SAT problem we are given a list of clauses $\phi = (C_1, C_2, \ldots, C_m)$, where each clause $C_j$ is a list of exactly 4 literals involving 4 distinct variables, say $(x, \overline{y}, \overline{w}, z)$. Given an assignment, we say that a clause is *equi-satisfied* if it contains *equal* numbers of TRUE and FALSE literals. For example, if $x, y, w$ are TRUE and $z$ is FALSE, $(x, \overline{y}, \overline{w}, z)$ would **not** be equi-satisfied because it contains one TRUE literal and three FALSE ones.

   Give a randomized approximation algorithm for finding an assignment to the variables that, in expectation, equi-satisfies a constant fraction $\rho \in [0, 1]$ of the clauses. Analyze what $\rho$ is exactly. Give a lower bound on the probability that your algorithm satisfies at least a $\rho/2$-fraction of the clauses, using Markov's inequality.

**Solution: Write all calculations and reasoning:** The algorithm: pick the truth assignment uniformly at random.

The analysis. Let $X_1, \ldots, X_m$ be indicators for whether clauses $C_1, \ldots, C_m$ are satisfied and $X = \sum_i X_i$. By linearity of expectation,

$$\begin{aligned}\mathbf{E}[X] = \sum_{i=1}^{m} \mathbf{E}[X_i] &= m \cdot \Pr(C_i \text{ is equi-satisfied}) \\ &= m \cdot \binom{4}{2} 2^{-4} \\ &= (3/8)m\end{aligned}$$

There are $\binom{4}{2} = 6$ ways to equi-satisfy a clause, and each occurs with probability $2^{-4} = 1/16$. Thus, $\rho = 3/8$.

To compute the probability that at least $\rho/2$ fraction of clauses are satisfied, we can use regular Markov's inequality to upper bound the fraction of unsatisfied clauses.

Let $Y$ denote the fraction of unsatisfied clauses. Then $\mathbf{E}[Y] = 1 - \rho = 5/8$. We want to upper bound the probability that more than $1 - \rho/2 = 13/16$ fraction of clauses are unsatisfied. By Markov,

$$\Pr[Y > 13/16] < \frac{5/8}{13/16} = \frac{10}{13}.$$

So, because $\Pr[Y > 13/16] < 10/13$, this implies with $\Pr[X \geq 3/16] \geq 3/13$, at least $\rho/2$ fraction of the clauses are satisied.

Alternatively, letting $Q = X/m$ (the fraction of satisfied clauses), we can compute $\Pr[Q \geq 3/16]$ directly by applying reverse Markov's:

$$\Pr[Q \geq 3/16] \geq \frac{\mathbf{E}[Q] - 3/16}{1 - 3/16} = \frac{3/8 - 3/16}{1 - 3/16} = \frac{3}{13}.$$

| $\rho$ | Probability at least $\rho/2$-fraction are satisfied |
|---|---|
| 3/8 | 3/13 |

3. SecureCo sells software to produce RSA keys. Every time you run SecureCo's `KeyGen()` procedure it randomly generates a tuple $(n, e, d)$, where $n = p \cdot q$, $de \equiv 1 \pmod{\phi(n)}$, and $p$ and $q$ are 512-bit primes. Unfortunately there is a subtle flaw in the code of `KeyGen()`: $p$ is a *fixed* 512-bit prime number, and $q$ is a *random* 512-bit prime number. (Although $p$ is fixed, you do not know what it is.)

   Suppose Alice uses `KeyGen()` to produce $(n, e, d)$ and publicly announces $(n, e)$, while Bob uses `KeyGen()` to produce $(n', e', d')$ and publicly announces $(n', e')$. Explain how to *efficiently* decode *any* encrypted message that you intercept, which was encrypted with $(n, e)$. You may assume $n \neq n'$. Recall that if $n = pq$, $\phi(n) = (p-1)(q-1)$.

**Solution:** We show how to decrypt any message $m$ that Alice sends.

**Step 1.** As $n$ and $n'$ are public knowledge, and they share the prime factor $p$, we first run $\mathsf{Euclid}(n, n')$ to determine $gcd(n, n') = p$ efficiently.

**Step 2.** Compute $q = n/p$.

**Step 3.** Now that we know $p$ and $q$, we compute $\phi(n) = (p-1)(q-1)$.

**Step 4.** We now know $e$ and $\phi(n)$, and can compute $d \equiv e^{-1} \pmod{\phi(n)}$ using extended Euclid's algorithm to compute inverses.

**Step 5.** For any secret message $m$ and any ciphertext $c = m^e \mod n$ intercepted, we can decrypt it as usual by $m = c^d \mod n$.

**UNIQNAME** (print clearly): ______________________________

**This page intentionally left blank.**

Be sure to write "answer continues on page 8" under your answer if you use this page.