Final Exam

- Do not open this exam booklet until you are directed to do so. Read all the instructions first.
- The quiz contains 6 multi-part problems. You have 180 minutes to earn 120 points.
- This quiz booklet contains 11 double-sided pages, including this one and a double-sided sheet of scratch paper; there should be 18 (numbered) pages of problems.
- This quiz is closed book. You may use three double sided Letter (8½” × 11”) or A4 crib sheet. No calculators or programmable devices are permitted. Cell phones must be put away.
- Write your solutions in the space provided. Extra scratch paper may be provided if you need more room, although your answer should fit in the given space.
- Do not waste time re-deriving facts that we have studied. It is sufficient to cite known results.
- Do not spend too much time on any one problem. Generally, a problem’s point value is an indication of how much time to spend on it.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Grade</th>
<th>Initials</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>120</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Name: ________________________________
Circle your recitation:

R01  R02  R03  R04  R05  R06
F10  F11  F12  F1   F2   F3
Joe  Joe  Khanh Khanh Emily Emily

R07  R08  R09  R10
F11  F12  F1   F2
Matt Matt Geoff Geoff
Problem 1. True or False, and Justify [42 points] (14 parts)

Circle T or F for each of the following statements, and briefly explain why. Your justification is worth more points than your true-or-false designation. If you need to make a reasonable assumption in order to answer a question (for example, if you need to assume that P ≠ NP), please state that assumption explicitly.

(a) T F [3 points] If problem A can be reduced to 3SAT via a deterministic polynomial-time reduction, and \( A \in \text{NP} \), then A is NP-complete.

(b) T F [3 points] Let \( G = (V, E) \) be a flow network, i.e., a weighted directed graph with a distinguished source vertex \( s \), a sink vertex \( t \), and non-negative capacity \( c(u, v) \) for every edge \( (u, v) \) in \( E \). Suppose you find an \( s-t \) cut \( C \) which has edges \( e_1, e_2, \ldots, e_k \) and a capacity \( f \). Suppose the value of the maximum \( s-t \) flow in \( G \) is \( f \).

Now let \( H \) be the flow network obtained by adding 1 to the capacity of each edge in \( C \). Then the value of the maximum \( s-t \) flow in \( H \) is \( f + k \).
(c) T F [3 points] Let $A$ and $B$ be optimization problems where it is known that $A$ reduces to $B$ in polynomial time. Additionally, it is known that there exists a polynomial-time 2-approximation for $B$. Then there must exist a polynomial-time 2-approximation for $A$.

(d) T F [3 points] There exists a polynomial-time 2-approximation algorithm for the Traveling Salesman Problem.
(e) T F [3 points] A dynamic programming algorithm that solves $\Theta(n^2)$ subproblems could run in $\omega(n^2)$ time.

(f) T F [3 points] If $A$ is a Monte Carlo program computing a predicate $f(x)$, and $B$ is a Las Vegas program computing a predicate $g(x)$, then

```
if A(x) then
    return B(x)
else
    return False
```

is a Monte Carlo program computing $(f(x) \land g(x))$. 
(g) T F [3 points] Dynamic programming programs require space at least proportional to the number of subproblems generated (in order to “memoize” the solution to each subproblem).

(h) T F [3 points] Let $H = \{h_i : \{1, 2, 3\} \rightarrow \{0, 1\}\}$ be a hash family defined as follows.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$h_3$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(For example, $h_1(3) = 0$.)
Then $H$ is a universal hash family.
(i) **T F** [3 points] If we use a max-queue instead of a min-queue in Kruskal’s MST algorithm, it will return the spanning tree of maximum total cost (instead of returning the spanning tree of minimum total cost). (Assume the input is a weighted connected undirected graph.)

(j) **T F** [3 points] Define a graph as being *tripartite* if its vertices can be partitioned into three sets $X_1, X_2, X_3$ such that no edge in the graph has both vertices in the same set. (That is, all edges are between vertices in different sets.) Then deciding whether a graph is tripartite can be done in polynomial time.
(k) T F [3 points] A randomized algorithm for a decision problem with one-sided-error and correctness probability $1/3$ (that is, if the answer is YES, it will always output YES, while if the answer is NO, it will output NO with probability $1/3$) can always be amplified to a correctness probability of $99\%$.

(l) T F [3 points] Let $B_0, B_1, B_2, \ldots$ be an infinite sequence of decision problems, where $B_0$ is known to be NP-hard and

$$B_i \leq_P B_{i+1} \text{ for all } i \geq 0.$$ 

Then it must be the case that $B_i$ is NP-hard for all $i \geq 0$. 
(m) T F  [3 points] Let $L$ be a decision problem. If there exists an interactive proof for $L$ where the verifier is deterministic, then $L \in NP$.

(n) T F  [3 points] Let $L$ be a decision problem. If there exists an interactive proof for $L$ where the prover runs in polynomial time, then $L \in P$. 
Problem 2. Short Answer [41 points] (9 parts)

Give brief, but complete, answers to the following questions.

(a) [7 points] Suppose that $n$ women check their coats at a concert. However, at the end of the night, the attendant has lost the claim checks and doesn’t know which coat belongs to whom. All of the women came dressed in black coats that were nearly identical, but of different sizes. The attendant can have a woman try a coat, and find out whether the coat fits (meaning it belongs to that woman), or the coat is too big, or the coat is too small. However, the attendant cannot compare the sizes of two coats directly, or compare the sizes of two women directly. Describe how the attendant can determine which coat belongs each woman in expected $O(n \log n)$ time. Give a brief analysis of the running time of your algorithm.
(b) [4 points] Let \( F_1, F_2, \ldots = 1, 1, 2, 3, 5, 8, \ldots \) denote the usual sequence of Fibonacci numbers (defined by \( F_1 = 1, F_2 = 1, \) and \( F_i = F_{i-1} + F_{i-2} \) for \( i > 2 \)).

Suppose that a file to be compressed contains \( k \) different symbols \( a_1, a_2, \ldots, a_k \) and that it contains \( F_i \) occurrences of \( a_i \) for each \( i \). Thus, if \( k = 4 \), the string has length 7 and contains 2 occurrences of \( a_3 \).

Assume the file is encoded with Huffman encoding. How many bits will be used to encode \( a_i \), as a function of \( i \) and/or \( k \)? State your answer concisely. You do not need to provide a proof.

(c) [4 points] As a final project for one of your other Course 6 classes, you have a massive program to run. After much effort, you are able to parallelize 90% of your code. The computer lab has two systems on which you could run your program:

- a cluster of 90 single-core computers each running at 1GHz, and
- a computer with 9 cores each running at 2GHz.

Which one should you choose to complete your project as quickly as possible?
(d) [5 points] Recall the clique problem from lecture: Given an undirected graph $G = (V, E)$ and a positive integer $k$, is there a subset $C$ of $V$ of size at least $k$ such that every pair of vertices in $C$ has an edge between them?

Ben Bitdiddle thinks he can solve the clique problem in polynomial time using linear programming.

- Let each variable in the linear program represent whether or not each vertex is a part of our clique. Add constraints stating that each of these variables must be nonnegative and at most one.
- We go through the graph $G$ and consider each pair of vertices. For every pair of vertices where there is not an edge in $G$, add a constraint stating that the sum of the variables corresponding to the endpoint vertices must be at most one. This ensures that both of them cannot be part of a clique if there is no edge between them.
- The objective function is the sum of the variables corresponding to the vertices. We wish to maximize this function.

Ben argues that the value of the optimum must be the size of the maximum-size clique in $G$, and we can then simply compare this value to $k$. Explain the flaw in Ben’s logic.
(e) [4 points] In a weighted connected undirected graph that might have negative-weight edges but no negative-weight cycles, how would you find a triple of distinct vertices \(x, y, z\) that minimizes \(f(x, y, z) = d(x, y) + d(y, z) + d(z, x)\) where \(d(u, v)\) is the length of the shortest path from \(u\) to \(v\)?

The running time of your algorithm should be \(O(n^3)\), where \(n\) is the number of vertices in the graph.

(f) [4 points] Suppose you are using RSA and you change your public key \((e, N)\) every so often, where \(N = pq\) is the product of your two large secret primes.

Why is it not a good idea to leave \(p\) the same and just replace \(q\) with a different secret prime \(q'\) (so your new \(N'\) is just \(pq'\))?
(g) [7 points] You are working at a hospital trying to diagnose patients; you may assume that each patient has exactly one disease. You know of \( m \) different diseases \( d_1, d_2, \ldots, d_m \). You have \( n \) different tests you can run (labeled \( T_1, T_2, \ldots, T_n \)), each of which comes up positive for some set of diseases and negative for other diseases. You would like to correctly diagnose all patients while giving them the minimum necessary number of tests—or, at least, close to the minimum number. Since you must send the tests to the lab for processing, all tests must be performed in parallel.

We say that a set of tests \( T \subseteq \{T_1, T_2, \ldots, T_n\} \) is comprehensive if, for every pair of diseases \((d_i, d_j)\), there is some test \( T_k \in T \) that distinguishes them—that is, it returns positive for one and negative for the other. The minimum-comprehensive-set problem (MCS) is the problem of finding a comprehensive set of tests of minimum cardinality. MCS is known to be NP-hard.

Describe a polynomial-time \( \alpha \)-approximation algorithm for the MCS problem, where \( \alpha = \ln(m(m-1)/2) \).

(h) [3 points] State the three properties a trapdoor function should have.
(i) [4 points] Suppose you are given a polynomial time algorithm DECISION-FACTOR that, given two integers $k$ and $n$, returns YES if $n$ has a prime factor less than $k$, and NO if $n$ does not. Give a polynomial time algorithm for computing a single prime factor of $n$. 
Problem 3. More Spy Games [9 points]

An enemy country, Elbonia, has \( n \) transmitter/receiver pairs \((t_i, r_i)\). You can model the position of each \( t_i \) and each \( r_i \) as a point in the plane. Enemy communications travel along the straight-line segment from \( t_i \) to \( r_i \). You can place eavesdrop units at any point in the plane, but a unit must be on the line segment from \( t_i \) to \( r_i \) in order to eavesdrop successfully. If you put a unit at the intersection of two such segments, that unit can eavesdrop on both transmitter/receiver pairs. Assume no three such segments intersect at a point.

Your intelligence agency has given you a list of the coordinates of all \( n \) enemy transmitter/receiver pairs. Briefly describe a polynomial time algorithm for finding the minimum number of eavesdrop units required to eavesdrop on all \( n \) transmitter/receiver pairs. (No proof needed.)
Problem 4. Almost Sorted [9 points]

A sequence $x_1, x_2, \ldots, x_n$ of real numbers is said to be sorted if

$$x_1 \leq x_2 \leq \cdots \leq x_n.$$ 

We say that $x_1, x_2, \ldots, x_n$ is D-almost-sorted for a non-negative real number $D$ if there exists another sequence $y_1, y_2, \ldots, y_n$ of real numbers such that $y_1, y_2, \ldots, y_n$ is sorted, and $\sum_i |x_i - y_i| \leq D$. (That is, by “shifting” values $x_i$ to new values $y_i$, such that the total amount of shifting is at most $D$, the new set of numbers is sorted.)

Describe concisely a polynomial-time algorithm which, given an input sequence $x_1, x_2, \ldots, x_n$ and a non-negative real number $D$, determines whether $x$ is $D$-almost-sorted.
Problem 5. Randomized 3-Coloring [8 points] (3 parts)

In an undirected graph \( G = (V, E) \), a coloring is a mapping \( c \) which assigns colors to vertices. We denote the color of vertex \( v \) by \( c(v) \).

We say a coloring \( c \) satisfies an edge \( e = (u, v) \) if \( c(u) \neq c(v) \) (that is, the endpoints of the edges are assigned different colors). Let the function \( s(c) \) count the number of satisfied edges under a coloring \( c \).

Define the 3-coloring optimization problem as follows: Given an undirected graph \( G = (V, E) \), output a coloring \( c \) such that \( c(v) \in \{R, W, B\} \) for all \( v \in V \), such that \( s(c) \) is maximized.

Here is one very simple randomized algorithm:

\[
\text{RANDOMIZED-COLOR}(G) \\
1 \quad \text{for each } v \in V \\
2 \quad \text{Pick a color uniformly at random in } \{R, W, B\} \\
3 \quad \text{Let } c(v) = \text{color picked} \\
4 \quad \text{return } c
\]

(a) [2 points] Let \( e \) be any edge. What is the probability that the coloring picked satisfies \( e \)?
(b) [2 points] What is the expected number of edges satisfied by the coloring produced by $c$? Justify.

(c) [4 points] Show that RANDOMIZED-COLOR is a polynomial-time randomized $(3/2)$-approximation algorithm for the 3-coloring optimization problem. That is, show that $E(s(c)) \geq (2/3)s(c^*)$ where $c^*$ is the optimal coloring.
Problem 6. Sublinear-Time Unimodal Testing [10 points]

We say that an array \( A[1..n] \) of real numbers is **unimodal** if there exists an integer \( k \) such that \( 1 \leq k \leq n \), \( A[1..k] \) is monotonically non-decreasing, and \( A[k..n] \) is monotonically non-increasing.

We say that \( A \) is \( \epsilon \)-far from being unimodal if you have to remove more than \( \epsilon n \) elements from \( A \) in order for the remaining sequence to be unimodal.

Give a sublinear-time property tester that, given an array \( A[1..n] \) of distinct real numbers:

- if \( A \) is unimodal, outputs \( YES \) with probability 1, and
- if \( A \) is \( \epsilon \)-far from being unimodal, outputs \( NO \) with probability at least \( 2/3 \).