Quiz 1

- Do not open this quiz booklet until you are directed to do so. Read all the instructions first.
- The quiz contains 6 multi-part problems. You have 80 minutes to earn 80 points.
- This quiz booklet contains 7 numbered pages of problems.
- This quiz is closed book. You may use one double sided Letter ($8\frac{1}{2}'' \times 11''$) 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 covered in lectures, homework or recitats. 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>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>80</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Name: __________________________

Circle your recitation:

R01 R02 R03 R04 R05 R06
F10 F11 F12 F1 F2 F3
Rafael Shaunak Lin Lin Yu Yu
R07 R08 R09 R10
F11 F12 F1 F2
Piotr Piotr Tom Tom
Problem 1. [1 points] Write your name on every page! Don’t forget the cover.

Problem 2. True or False, and Justify [12 points] (4 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.

(a) T F A dynamic program’s sub-problem graph (where there is an edge from sub-problem A to sub-problem B iff solving A requires a solution to B) must be a tree.
   Justification:

(b) T F Let h be a function chosen uniformly at random from a universal hash family that maps keys from the domain \( \{1, \ldots, n\} \) to the range \( \{1, \ldots, m\} \). Let \( a, b, c \in \{1, \ldots, n\} \) such that \( a \neq b, a \neq c \) and \( b \neq c \).
   The probability that \( h(a) = h(b) = h(c) \) is at most \( 1/m \).
   Justification:
(c) T F You are given a weighted graph \( G = (V, E, W) \) such that (i) the weights \( W \) are non-negative and (ii) \( |E| = |V|^{1.5} \). Your goal is to compute all-pairs shortest paths in \( G \).

Among all algorithms seen in the class, Floyd-Warshall algorithm achieves the lowest running time for this problem.

Justification:

(d) T F Assume that we have a randomized primality testing algorithm that given a number \( N \) does the following (i) if the number is prime, it always reports YES and (ii) if the number is composite, it outputs NO with probability at least \( 3/4 \).

In order to ensure that the algorithm reports NO in case (ii) with probability at least \( 1 - 1/k \), we need to repeat it \( k \) times.

Justification:
Problem 3. Short Answer [20 points] (4 parts)

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

(a) Solve the recurrence \( T(n) = 8T(n/2) + n^3 \lg n \), i.e. provide a function \( f(n) \) such that \( T(n) = \Theta(f(n)) \). As usual, assume that \( T(n) = O(1) \) for \( n \leq 2 \).

(b) You are given a set \( P \) of \( n \) points in the plane \( (x_i, y_i) \) for \( i = 1, 2, ..., n \). Give an efficient randomized algorithm for determining if there exist three points in \( P \) that are co-linear, i.e., lie on the same line.

For full credit your algorithm should run in expected time \( O(n^2) \). You can assume that all arithmetic is exact.
(c) Give an efficient randomized Monte Carlo algorithm that, given two \( n \times n \) matrices \( A \) and \( B \), checks whether \( AB = BA \), i.e., whether the matrices commute.

(d) Recall that a heap performs insertion and deletion in worst case \( O(\lg n) \). Argue that (without changing the data structure), a heap can perform insertion in amortized \( O(\lg n) \) and deletion in amortized \( O(1) \).
Problem 4. A tale of two programs [12 points]

In this problem we assume that all arithmetic operations are modulo 2.

You are given an access to two programs $P_1$ and $P_2$. Both $P_1$ and $P_2$ take a binary vector $x$ of length $n$ as an input, and produce a binary vector of length $n$ as an output. Both of them are supposed to compute $A_1x$ for some binary $n \times n$ matrix $A_1$. Alas, only one of them ($P_1$) does it correctly; the other program ($P_2$) computes $A_2x$, where $A_1$ and $A_2$ differ in one entry.

Show how to identify the entry on which $A_1$ and $A_2$ differ using a small number of calls to programs $P_1$ and $P_2$. Note that you do not have any direct access to the matrices $A_1$ and $A_2$. 
Problem 5. Optimal coins [15 points]

You are given a sequence of \( n \) coins: \( C_1, \ldots, C_n \).

Each coin is biased: you are told that the \( i \)-th coin \( C_i \) has probability \( p_i \) of coming up heads (call this "1") and probability \( q_i = 1 - p_i \) of coming up tails (call this "0"). You are given the values of \( p_i \) (and of \( q_i \)) for \( i = 1, 2, \ldots, n \).

Next you are given a sequence \( G_j, j = 1, 2, \ldots, k \) where each \( G_j \) is either 1 or 0. Here \( 1 \leq k \leq n \).

You are asked to efficiently find an increasing subsequence \( i_1, i_2, \ldots, i_k \) of \( 1, 2, \ldots, n \) such that when all coins are flipped independently

\[
\text{Prob}(C_{i_j} = G_j \text{ for all } j = 1, 2, \ldots, k)
\]

is maximized.

Give an efficient algorithm for finding such a subsequence that maximizes your chance of obtaining the given goal sequence \( G_1, G_2, \ldots, G_k \) for the indices your algorithm produces.

Example: Consider a sequence of probabilities \( p_1, \ldots, p_4 \) equal to 0.7, 0.1, 0.2, 0.8, and a sequence \( G_1, G_2 \) of coins equal to 0, 1. In this case, the optimal solution \( i_1, i_2 \) can be seen to be equal to 2, 4, since the probability

\[
\text{Prob}(C_2 = 0 \text{ and } C_4 = 1) = (1 - 0.1) \cdot 0.8 = 0.72
\]

is maximized.
Problem 6. Searching Noisy Sequences, generalized [20 points]

You are given a sequence of symbols \( t = t[0] \ldots t[n-1] \), and a pattern \( p = p[0] \ldots p[m-1] \), where the symbols \( p[i], t[i] \) come from the set \( \{0 \ldots L\} \). The goal is to match \( p \) and \( t \). That is, for a specified symbol scoring function \( sc(a, b) \) that returns the score of a match between symbols \( a \) and \( b \), the goal is to find the match score

\[
\text{score}(s) = \sum_{i=0}^{m-1} sc(t[s+i], p[i])
\]

for all shifts \( s = 0 \ldots n - m \).

In pset 2 you have seen an algorithm (for a closely related problem) that works for the binary alphabet, i.e., \( L = 2 \). Our goal is to extend that algorithm to make it work for larger alphabet sizes.

We consider a scoring function \( sc \) such that \( sc(a, b) = 0 \) if \( a = b \) and \( sc(a, b) = 1 \) otherwise. That is, \( \text{score}(s) \) computes the number of mismatches between \( p \) (shifted by \( s \)) and \( t \).

(a) Give a deterministic algorithm that computes \( \text{score}(s) \) for all shifts \( s \). For full credit your algorithm should run in time \( O(Ln \log n) \).
(b) Give a randomized algorithm that for each $s$ provides an estimate $E(s)$ such that the expected value of $E(s)$ is equal to $\text{score}(s)/2$. Your algorithm should run in time $O(n \log n)$.

**Hint:** Consider hashing the alphabet into a smaller range, and using the algorithm from part (a).