This is a four hour exam. There are six questions, worth 10 points each. Answer all questions and all their subparts.

Please print your name and SID on the front of the envelope only (not on the exam booklets). Please answer each question in a separate booklet, and number each booklet according to the question.

Read the questions carefully. Keep your answers legible, and brief but precise. Assume standard results.

You must prove correctness of your algorithm and prove its time bound if asked. The algorithm can be written in plain English (preferred) or as a pseudo-code.

The graphs are undirected in Problems 1, 6 and directed in Problems 3, 4.

Best of luck!
Problem 1 (Graphs are undirected)
Let $T(V, E)$ be a tree and $w : V \rightarrow \mathbb{R}^+$ be a non-negative weight function on the vertices. The weight of a subset of vertices is the sum of weights of the vertices in it. Design a polynomial time algorithm to find an independent set with the maximum total weight.

Note: A tree is a connected graph with no cycles. An independent set is a subset of vertices such that there is no edge between any pair of vertices in this subset.

Problem 2
Given an array $A[1..n]$ with $n$ integers, we want to locate for each $i = 1, \ldots, n$, the smallest index $b(i)$ such that $A[i] < A[b(i)]$ and $i < b(i)$. If no such index exists, let $b(i) = 0$. We can view the function $b(i)$ as another array $B[1..n]$.

E.g., let $n = 7$ with $A$ given by

<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A[i]$</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>2</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B[i]$</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Design an $O(n)$ time algorithm that given an array $A$ computes the corresponding array $B$.

Hint: Use a stack.
Problem 3 (Graphs are directed)

Let \( G(V, E) \) be a directed graph. A vertex \( r \in V \) is called a root of \( G \) if every vertex in \( V \) is reachable from \( r \) via a (directed) path in \( G \).

Design an algorithm that finds a root of \( G \) if one exists, and otherwise outputs “NO ROOT”. Assuming \( G \) is represented using adjacency lists, your algorithm should run in time \( O(|V| + |E|) \).

*Hint: You could first consider the case when the graph is acyclic.*

Problem 4 (Graphs are directed)

The goal of this problem is to travel from home to a store, purchase a gift, and then get back home, at minimal cost.

Let us model this problem using a directed graph. Let \( G(V, E) \) be a directed, weighted graph, with nonnegative edge weights \( w : E \to \mathbb{R}^+ \). The weight of an edge represents the cost of traversing that edge. Each vertex \( v \in V \) also has an associated cost \( c(v) \in \mathbb{R}^+ \) which represents the cost of purchasing the desired gift at that location.

Starting from “home base” \( h \in V \), the goal is to find a location \( v \in V \) where the gift can be purchased, along with a path \( p \) from \( h \) to \( v \) and back from \( v \) to \( h \). The cost of such a solution is the cost \( c(v) \) of the location \( v \) plus the weight \( w(p) \) of the path \( p \) (i.e., the sum of edge weights along the path \( p \)).

Design an algorithm that on input \( G(V, E) \), including edge weights \( w(\cdot) \) and costs \( c(\cdot) \), and home base \( h \in V \), finds a minimal cost solution. Assuming \( G \) is represented using adjacency lists, your algorithm should run in time \( O((|V| + |E|) \log |V|) \).

Problem 5

The 4LIN problem consists of \( n \) Boolean (i.e. \( \{0,1\} \)-valued) variables \( x_1, x_2, \ldots, x_n \) and \( m \) equations where each equation is of the form:

\[
x_i \oplus x_j \oplus x_k \oplus x_\ell = 1, \quad 1 \leq i < j < k < \ell \leq n.
\]

Here \( \oplus \) denotes the xor operation.

1. If the variables in some fixed equation are assigned \( \{0,1\} \) values uniformly and independently, what is the probability that the equation is satisfied? Justify.

2. Show that there is an assignment to (all the \( n \)) variables that satisfies at least \( \frac{m}{2} \) equations.

3. Now assume that there exists an assignment that satisfies all the equations. Design a polynomial time algorithm to find such an assignment (i.e. one that satisfies all the equations). What is the complexity?

*Hint: Part 3 is not necessarily dependent on the previous parts. Think of linear systems.*
Problem 6 (Graphs are undirected)

A forest is a graph with no cycles. For a graph $G(V, E)$ and a subset $U \subseteq V$ of its vertices, let $G|_U$ denote the induced subgraph of $G$ on the set of vertices $U$ (i.e. the graph with the vertex set $U$ and edges that are precisely the edges of $G$ with both the endpoints in $U$). Show that the following problem, called FOREST, is NP-complete:

$$\text{FOREST} = \{(G(V, E), k) \mid G(V, E) \text{ is a graph and } \exists U \subseteq V, |U| = k \text{ such that } G|_U \text{ is a forest}\}.$$

Hint: You may use a reduction from the INDEPENDENT SET problem. Recall that

$$\text{INDEPENDENT SET} = \{(G'(V', E'), k') \mid G'(V', E') \text{ is a graph that has an independent set of size at least } k'\}.$$

Such a reduction maps an instance $(G'(V', E'), k')$ of the INDEPENDENT SET problem to an instance $(G(V, E), k)$ of the FOREST problem such that $G'$ has an independent set of size (at least) $k'$ if and only if $G$ has a forest of size (at least) $k$. 