1. Let $P$ be a set of $n$ points in the plane. Recall from the midterm that the **staircase of $P$** is the set of all points in the plane that have at least one point in $P$ both above and to the right.

(a) Describe and analyze a data structure that stores the staircase of a set of points, and an algorithm $\text{ABOVE}(x,y)$ that returns $\text{TRUE}$ if the point $(x, y)$ is above the staircase, or $\text{FALSE}$ otherwise. Your data structure should use $O(n)$ space, and your $\text{ABOVE}$? algorithm should run in $O(\log n)$ time.

(b) Describe and analyze a data structure that maintains the staircase of a set of points as new points are inserted. Specifically, your data structure should support a function $\text{INSERT}(x, y)$ that adds the point $(x, y)$ to the underlying point set and returns $\text{TRUE}$ or $\text{FALSE}$ to indicate whether the staircase of the set has changed. Your data structure should use $O(n)$ space, and your $\text{INSERT}$ algorithm should run in $O(\log n)$ amortized time.

2. An **ordered stack** is a data structure that stores a sequence of items and supports the following operations.

- $\text{ORDEREDPUSH}(x)$ removes all items smaller than $x$ from the beginning of the sequence and then adds $x$ to the beginning of the sequence.
- $\text{POP}$ deletes and returns the first item in the sequence (or $\text{NULL}$ if the sequence is empty).

Suppose we implement an ordered stack with a simple linked list, using the obvious $\text{ORDEREDPUSH}$ and $\text{POP}$ algorithms. Prove that if we start with an empty data structure, the amortized cost of each $\text{ORDEREDPUSH}$ or $\text{POP}$ operation is $O(1)$.

3. Consider the following solution for the union-find problem, called **union-by-weight**. Each set leader $\bar{x}$ stores the number of elements of its set in the field $\text{weight}(\bar{x})$. Whenever we $\text{UNION}$ two sets, the leader of the smaller set becomes a new child of the leader of the larger set (breaking ties arbitrarily).

\[
\begin{align*}
\text{MAKESET}(x) & \quad \text{parent}(x) \leftarrow x \\
& \quad \text{weight}(x) \leftarrow 1 \\
\text{FIND}(x) & \quad \text{while } x \neq \text{parent}(x) \\
& \quad x \leftarrow \text{parent}(x) \\
& \quad \text{return } x \\
\text{UNION}(x, y) & \quad \bar{x} \leftarrow \text{FIND}(x) \\
& \quad \bar{y} \leftarrow \text{FIND}(y) \\
& \quad \text{if weight}(\bar{x}) > \text{weight}(\bar{y}) \\
& \quad \quad \text{parent}(\bar{y}) \leftarrow \bar{x} \\
& \quad \quad \text{weight}(\bar{x}) \leftarrow \text{weight}(\bar{x}) + \text{weight}(\bar{y}) \\
& \quad \quad \text{else} \\
& \quad \quad \text{parent}(\bar{x}) \leftarrow \bar{y} \\
& \quad \quad \text{weight}(\bar{x}) \leftarrow \text{weight}(\bar{x}) + \text{weight}(\bar{y})
\end{align*}
\]

Prove that if we use union-by-weight, the worst-case running time of $\text{FIND}(x)$ is $O(\log n)$, where $n$ is the cardinality of the set containing $x$. 

\[\]