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 ABOVE?(x, y) that returns TRUE if the point (x, y) is above the staircase, or FALSE otherwise. Your data structure should use $O(n)$ space, and your 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 INSERT(x, y) that adds the point (x, y) to the underlying point set and returns TRUE or FALSE to indicate whether the staircase of the set has changed. Your data structure should use $O(n)$ space, and your 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.

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

   Suppose we implement an ordered stack with a simple linked list, using the obvious ORDEREDPUSH and POP algorithms. Prove that if we start with an empty data structure, the amortized cost of each ORDEREDPUSH or 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 weight$(\bar{x})$. Whenever we UNION two sets, the leader of the *smaller* set becomes a new child of the leader of the *larger* set (breaking ties arbitrarily).

```
MAKESET(x)
    parent(x) ← x
    weight(x) ← 1

FIND(x)
    while x ≠ parent(x)
        x ← parent(x)
    return x

UNION(x, y)
    x̄ ← FIND(x)
    ȳ ← FIND(y)
    if weight(x̄) > weight(ȳ)
        parent(ȳ) ← x̄
        weight(x̄) ← weight(x̄) + weight(ȳ)
    else
        parent(x̄) ← ȳ
        weight(x̄) ← weight(x̄) + weight(ȳ)
```

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