Note: For ease of reference I will try to use e.g. HW5.3 to refer to Problem set 5’s 3rd assigned problem, as opposed to e.g. K-T 3.4 which would be Chapter 3, exercise 4 from Kleinberg-Tardos.

HW1.0. [no points] Solve, but do not turn in: Kleinberg-Tardos, Chapter 1, Exercises 2 and 4

HW1.1. [10 points] Here we consider the Gale-Shapley algorithm (in the version where men propose). Recall that in a given iteration of the While loop, the man who will propose is (in general) not fully determined by the algorithm.

Show that for all $N > 1$, there is a list of $N$ men’s and $N$ women’s preferences, such that Gale-Shapley will require $\Omega(N^2)$ proposals$^1$ to terminate in a stable marriage, regardless of the selection of single men to propose at each phase.

Hint: try preference lists with a simple structure.

---

$^1$Recall this notation means $\geq cN^2$ for some constant $c > 0$ independent of $N$. 
HW1.2. [10 points] There is a very simple algorithm, which I’ll call the *Laissez-Faire (LF)* algorithm, to attempt to produce a stable opposite-sex matching between $N$ men and $N$ women. Here is the algorithm:

1. Let $\mathcal{M}$ be an arbitrary initial matching between men and women;

2. If there is a potential eloping pair with respect to $\mathcal{M}$, each of which prefers each other over their current match, let $(m, w)$ be any such pair—chosen arbitrarily, if there’s more than one. (If no such pair exists, **halt** with output $\mathcal{M}$.)

3. Modify $\mathcal{M}$ by having the chosen $(m, w)$ become matched; also, if $m', w'$ are the pair of people they just ditched, then $m', w'$ are re-matched with each other.


Your job is to show that this algorithm is capable of running forever, without producing a stable matching—at least, this can happen for some value of $N$, on some collection of individual preferences, and for some sequence of possible choices made by the algorithm (which, note, is not fully deterministic).

*Hint:* there are two things you must produce: a description of preferences, and a “story,” or sequence of elopement events, describing how the algorithm can run forever. It might be easier to tell the story first, and then design the preferences.

Note the difference in structure between this statement and that of the previous problem. That problem asserted that for some preference lists, a certain algorithm behavior is *inevitable*; this problem asserts that for some pref. lists, a certain algorithm behavior is *possible.*
HW1.3. [10 points] A new type of Pokémon called Scaremancer has been discovered! Each Scaremancer has an associated level, which is a positive integer. If a Scaremancer becomes frightened, it will disappear in a puff of green smoke and be replaced by some number of new Scaremanders. Any finite number of new Scaremanders may appear, but they will all be of level strictly less than the one from which they spawned. So if a level-1 Scaremancer is frightened, it disappears with no new ones to replace it.

Scaremanders have no other way of reproducing and cannot increase their level. A finite initial population is discovered, confined to live in a spooky forest where at least one Scaremancer gets frightened every day that the group persists there.

Prove that this population must eventually drop to zero.

*Hint:* try some kind of induction (or, double induction). Note, this problem’s result implies that a broad class of recursive algorithms must eventually halt.