(2 points) 1. Design an algorithm that detects an occurrence of a cube (a word of the form xxx) in a string of length n in $\mathcal{O}(n)$ time by generalizing the algorithm for squares. Do you see how to further generalize the algorithm to detect all k-powers in $\mathcal{O}(kn)$ time?

(2 points) 2. Prove that a string of length n contains at most $2n$ **distinct** squares. The key insight is to choose the rightmost occurrence of every distinct square. Then, argue that any position in the string can be the rightmost position of at most two such occurrences.

(2 points) 3. Construct an infinite family of strings of length n containing at least $n - o(n)$ **distinct** squares. You might find strings $q(i) = 0^{i+1}10^i10^{i+1}1$ helpful in your construction.

4. Show how to construct, given a self-referential LZ parse consisting of z phrases describing $w[1..N]$, a non-self-referential LZ parse consisting of $\mathcal{O}(z \log(N/z))$ phrases and describing the same string.

5. Recall that in LZ parsing we always choose the next phrase to be as long as possible. Argue formally that this is indeed optimal, that is, results in the smallest number of phrases.

6. Is it true that LZ78 parsing choosing the next phrase to be as long as possible is the best possible choice? If not, construct a string such that the number of phrases obtained when choosing the next phrase greedily differs significantly from the number of phrases in an optimal parsing.