Reminder: You may work in groups and use outside sources. But, you must write up solutions in your own words and properly reference your sources for each problem. This includes listing your collaborators and properly citing any sources you use. Solutions to each problem must be electronically typeset and submitted online via Blackboard. Instructions appear in the E-Homework Guide: http://www.cse.wustl.edu/~bjuba/cse347/s17/ehomework/ For all problems in this course, your solutions should provide a proof of both correctness and a running time bound, unless the problem explicitly states that this is not necessary.

1. Kleinberg & Tardos – Chapter 13, exercise 8

2. Kleinberg & Tardos – Chapter 13, exercise 15

3. In lecture, we saw how to choose hash functions that satisfy a stronger property than universality, pairwise independence: that is, for a hash function $h_a$ chosen uniformly at random from the family $\mathcal{H}$, for any two elements $x$ and $y$ in the universe and any two values $w$ and $z$ in the range $(0, \ldots, m - 1)$, $\Pr_a[h_a(x) = w \text{ and } h_a(y) = z] = 1/m^2$.

Here is an application of pairwise independent hash functions in message authentication: suppose that Alice and Bob secretly agree on a member of $h_a \in \mathcal{H}$. Then, when Alice wishes to send a message $x$ to Bob, she sends $x$ attached to $w = h_a(x)$. Bob then checks that the pair $(x, w)$ he receives satisfies $w = h_a(x)$, and if so, accepts $x$ as genuine. (Otherwise he rejects the message.)

Show that for a suitable choice of $m$, no adversary who intercepts $(x, w)$ can corrupt it to some $(y, z)$ for $y \neq x$ that Bob will accept as genuine except with probability at most $\delta$, even if the adversary knows what family $\mathcal{H}$ Alice and Bob are using and can spend unlimited computational resources crafting $(y, z)$ from $(x, w)$. How large does $m$ (and hence, the tag $w$) need to be?

4. Show how to construct a dictionary data structure that, with probability $1 - \delta$, supports lookup and delete in $O(1)$ time and inserts in $O(1)$ amortized time per insert when the number of inserts $n$ is not known in advance. Your data structure should only require an amount of memory bounded by $O(\frac{1}{\delta} n^2 \log^2 n)$. (Note that by convention, $\log^2 n = (\log n)^2$; not $\log \log n$.)

For this problem, you may assume that you have a function that on input $k$ returns a $k$-bit prime number (i.e., between $2^k$ and $2^{k+1}$) and runs in time polynomial in $k$. You may also assume that allocating any amount of memory initialized to NULL can be done in $O(1)$ time. (Note: this last assumption can be removed.)