Problem Set 6—Due Friday, May 12

Problem 1 Remember the universal hash function that we looked at in class: Letting each m_i be a byte, interpreted as a number between 0 and 255, letting p be the prime number $2^{16} - 15$, and letting $k \in \{0, \ldots, p-1\}$, we set

$$h_k(m_n \ldots m_1 m_0) = (m_n k^n + \cdots + m_1 k + m_0) \bmod p$$

Part A. Show that this hash function is *not* ϵ-almost univeral (for a desirable value of ϵ) if you think of the domain as strings of 128 or fewer bytes. *Hint: give a pair of strings that always collides.*

Part B. Slighlty modify the hash-function family so as to fix the "defect" of Part A.

Part C. On many computers, like the Intel x86 family, it is slow to use the native mod function of the instruction set architecture. Let $s, k \in \{0, \ldots, p-1\}$, $m_i \in \{0, \ldots, 255\}$. Describe a fast way to compute $s = (sk + m_i) \bmod p$ where you do not use integer division or modular remainder.

Problem 2 Page 183, Exercise 9.4–3.

Problem 3 In the Stable Roommate Problem one is given an even number n and n "preference lists", $L_1, \ldots, L_n$. Each preference list L_i is a permutation on $\{1, \ldots, n\} - \{i\}$. We say that person i *prefers* $L_i(j)$ to $L_i(k)$ if $j < k$. A *stable matching* of such an instance is a partition of $\{1, \ldots, n\}$ into two-element subsets, S, such that $\{x, x'\}, \{y, y'\} \in S$ implies x prefers x' to y or y prefers y' to x. Prove the following: sometimes there doesn't exist a stable matching.

Problem 4 The greedy algorithm for making change using pennies, nickels, dimes, and quarters is very straightforward: for any number of cents, n:

while $(n > 25)$ $\{$GiveQuarter(); $n \leftarrow n - 25\}$
while $(n > 10)$ $\{$GiveDime(); $n \leftarrow n - 10\}$
while $(n > 5)$ $\{$GiveNickel(); $n \leftarrow n - 5\}$
while $(n > 1)$ $\{$GivePenny(); $n \leftarrow n - 1\}$

Prove that the above greedy algorithm always dispenses the minimum number of coins to make n cents.