Final exam

CS70, Blum/Wagner, 19 May 2001

This is a CLOSED BOOK examination. One page of notes is permitted. Calculators are permitted. Do all your work on the pages of this examination. Give reasons for all your answers.

Be sure to do all work for each problem on the pages provided for that problem.
Print your full name on every page of the examination.

PRINT your full name here: ______________________, ______________________
(last) (first)

SIGN your name: ____

Problem 1. (Repetitions) [10 points]
We say that a string of bits has k *quadruply-repeated ones* if there are k positions where four consecutive 1's appear in a row. For example, the string 0100111110 has two quadruply-repeated ones.

What is the expected number of quadruply-repeated ones in a random n-bit string, when $n \geq 3$ and all n-bit strings are equally likely? Justify your answer.

Problem 2. (There's gold in them thar hills!) [20 points]
You have n gold coins. One of them is fraudulent, and the rest are good.

Unfortunately, you don't know which one is fraudulent: they all look alike. Good coins weigh one ounce, but the bad coin has a different weight, and this is the only way to identify which one is bad. Fortunately, you have a balance scale and an endless supply of known-good coins (lucky you!).

Consider the following recursive algorithm for finding the bad coin from a set S of n coins:

```
FindBad(S):
    If S has just one coin:
        Return that coin.
    Let n be the number of coins in S.
    If n is not a multiple of 3:
        Add enough known-good coins to S to bring the size up to a multiple of 3.
        Return FindBad(S).
    Randomly divide S into three piles A, B, C each of size n/3
    (so that all ways to evenly divide n coins into three piles are equally likely).
    Weigh pile A against pile B.
    If the scale balances:
        Return FindBad(C).
    Merge piles A and B into a combined pile D.
    Return FindBad(D).
```

Let a_n be the expected number of weighings used by the above algorithm to find the bad coin from a set of n coins.

(a) [5 points] Fill in the following. Show your work.

$a_1 =$ ______________.

$a_2 =$ ______________.

$a_3 =$ ______________.

$a_4 =$ ______________.

(b) [15 points] Let $\lg n$ refer to the base-2 logarithm of n. Prove: $a_n \leq 3 \lg n$ for all $n \geq 1$.

[Hint: You might consider examining a_m, where m is the smallest multiple of 3 that is $\geq n$. You may assume without proof that, for this choice, $2m/3 < n$ holds for all $n > 4$. Also, you can freely use the following facts about logarithms: $\lg(a) + \lg(b) = \lg(ab)$, $m \lg a = \lg(a^m)$, $\lg 2 = 1$, $\lg 4 = 2$, and $\lg x \leq 0$ when $0 < x \leq 1$.]

If necessary, you may state some small assumptions as needed to complete your proof and we will give an appropriate amount of partial credit.

Problem 3. (Codebreaking) [20 points]
You've forgotten your code to your answering machine. All you remember is that it is a two-digit code, and the two digits aren't the same.

You have a clever idea: You'll enter a string that contains all possible pairs of distinct digits. For instance, if answering machine codes were made up from the digits 1,2,3, you could enter the string 121321231 and be sure of getting access, because every pair of distinct digits appears somewhere in this string (for instance, 32 appears at the fourth position). However, this is not shortest string with this property. To ease your weary fingers, you want to find a string with this property that is as short as possible.

Let ℓ_n be the length of such a shortest string if the possible digits are $1, 2, \ldots, n$.

(a) [3 points] What is ℓ_3? Justify your answer. [Hint: consider the directed graph $G = (V, E)$ with $V = \{1, 2, 3\}$ and $E = \{(i, j) : i \neq j\}$.]

(b) [1 point] What is ℓ_4?

(c) [10 points] What is ℓ_n? Prove your answer.

(d) [6 points] Give an algorithm for finding a string of length ℓ_n with the above property. Try to avoid making it unnecessarily inefficient (try to make it run in time polynomial in n).

Problem 4. (A proof) [10 points]
Consider the following result, first proved many centuries ago.

Theorem 1 (Euclid). *There exist infinitely many primes.*

Proof. Assume to the contrary that there exist finitely many primes. Let these primes (in increasing order) be $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $\ldots$, p_k. Let $q_k = p_1 p_2 p_3 \cdots p_k + 1$. Note that q_k is a new number not in the list of primes $p_1, \ldots, p_k$. At the same time, it is not divisible by p_i for any i, since $q_k \equiv p_1 p_2 p_3 \cdots p_k + 1 \equiv 1 \pmod{p_i}$, which would mean that q_k is a new prime different from $p_1, \ldots, p_k$, which is a contradiction. This completes the proof. □

Let $p_1, \ldots, p_k$ represent the first k primes. Are we guaranteed that $p_1 p_2 p_3 \cdots p_k + 1$ is always prime for all $k \geq 1$? Justify your answer.

Problem 5. (Error correcting codes) [15 points]
Let $m_1 < m_2 < \ldots < m_k$ be k pairwise relatively prime (positive) integers. The term *pairwise relatively prime* means that $\gcd(m_i, m_j) = 1$ for all distinct (unequal) $i, j \in \{1, ..., k\}$. Here, as usual, gcd = greatest common divisor.

(a) [2 points] Are 13,14,15,17 pairwise relatively prime? ____________
Are 15,17,19,21 pairwise relatively prime? ____________

For positive integers a, m, we computer scientists define: $a \bmod m$ = remainder upon dividing a by $m = a - m \cdot \text{floor}(a/m)$. For example, $7 \bmod 3 = 1$, and $23 \bmod 5 = 3$.

(b) [1 point] $42 \bmod 11 =$ ________________________.

Let $S = \{0, 1, \ldots, s\}$ be any finite set of consecutive nonnegative integers starting from 0. A *Chinese remainder code* encodes an integer $x \in S$ as a pair of k-tuples:

$$E(x) = [\langle x \bmod m_1, ..., x \bmod m_k \rangle, \langle b_1, ..., b_k \rangle]$$

where b_i is the parity of the bit string $x \bmod m_i$. Recall that the *parity* of a bit string is the sum modulo 2 of the bits of the string. For instance, the parity of 01101 is 1.

(c) [2 points] In the special case where $S = \{0, 1, 2, \ldots, 63\}$ is the set of all nonnegative 6-bit integers, and m_1, m_2, m_3 are the three pairwise relatively prime numbers $m_1 = 9 < m_2 = 10 < m_3 = 11$, we have (fill in the blanks):

$$E(27) = [\langle 0, 7, \_\_\_\_\_\_\_\_ \rangle, \langle 0, 1, \_\_\_\_\_\_\_\_ \rangle]$$

(d) [2 points] In general, for any given pairwise relatively prime (positive) integers $m_1, \ldots, m_k$, what is the largest allowable value of s (the largest element in S) to ensure that the function E is 1:1 on S? In other words, you should find the largest value of s that ensures that

$$\text{for all } x, y \in S, \quad [E(x) = E(y) \Rightarrow x = y].$$

Give your answer as a function of $m_1, \ldots, m_k$:

$s =$ ____________________________________.

In Chinese remainder codes, the parity bits are used to detect errors in each of the k entries of $\langle x \bmod m_1, ..., x \bmod m_k\rangle$.

As before, let $S = \{0, 1, 2, \ldots, 63\}$, and let m_1, m_2, m_3 be the three pairwise relatively prime numbers $m_1 = 9 < m_2 = 10 < m_3 = 11$.

(e) [4 points] Suppose that during transmission, at most one bit of $E(x)$ is (mistakenly) flipped, and that what is received is $[\langle 8, 5, 6\rangle, \langle 1, 1, 0\rangle]$. How should this be decrypted?

$x =$ ____.

(f) [1 point] Is it possible from $[\langle 8, 5, 6\rangle, \langle 1, 1, 0\rangle]$ to recover not only the original x but also the encoded string, $E(x)$? ______________
If so, what do you get for $E(x)$?

$E(x) =$ ____.

In general, the above code detects and corrects any single (1-bit) error.

Most algorithms for decoding Chinese remainder codes make calls to a simple efficient subroutine that computes:

INVERSE$(a_1, \ldots, a_k; m_1, \ldots, m_k)$:
Input: Pairwise relatively prime positive integers $m_1, \ldots, m_k$, and nonnegative integers $a_1, \ldots, a_k$ such that $a_i < m_i$ for each i.
Output: The unique integer x in $\{0, \ldots, m-1\}$ such that $x \bmod m_1 = a_1, \ldots, x \bmod m_k = a_k$, where $m = m_1 \cdots m_k$.

(g) [3 points] For example, INVERSE$(1, 2, 3; 9, 10, 11) = 982$. Show that this answer is correct.

Extra Credit. (More error correcting codes)
This question, a continuation to the previous question, is optional and will count for extra credit.

Suppose we would like to detect and correct up to two (1-bit) errors in a set S, where $S = \{0, 1, \ldots, s\}$. Let $k = 4$. Let $m_1 < \ldots < m_k$ be pairwise relatively prime positive integers such that $m_1 m_2 > s$.

(a) Give an algorithm to decode the received transmission of x in case there are at most two (single-bit) errors. Comment your algorithm so that it's clear not only what it's doing, but also why. You may call the INVERSE() subroutine of question 5(g) in your algorithm.

As usual, bit errors may be in an $x \bmod m_i$ or a b_i or both. You might want to think about parts (b), (c), and (d) before answering part (a).

(b) Justify briefly that your algorithm works in the case that there are exactly two 1-bit errors, and no parity bits are in error.

(c) Justify briefly that your algorithm works in the case that there are exactly two 1-bit errors, and two parity bits are in error.

(d) Justify briefly that your algorithm works in the case that there are two 1-bit errors, and one of these errors occurs in a parity bit b_i and the other in some bit of its associated integer $a_i = x \bmod m_i$.