Math 203 Contemporary Mathematics

Topics for the first quiz: Check Digit Systems and Modular Arithmetic

To do:
Compute the check digit from the description of a system.
Recover a missing digit knowing the remainder of the digits (detect single digit errors).
Describe how a system can or cannot detect one of the two typical errors made in entering a number with a check digit: single digit errors and the transposition of two adjacent digits.

Main tools: solve $a \equiv r \pmod{m}$ for r (i.e., find remainders); solve $cx + r \equiv 0 \pmod{m}$ for x.

A unifying language for check digit systems: modular arithmetic.

Starting point: quotients and remainders. Given two whole numbers a and m, there are unique numbers q (the quotient) and r (the remainder) with $0 \leq r \leq m-1$ satisfying $a = q \cdot m + r$. Two ways to compute: $\frac{a}{m} = q + \frac{r}{m}$, so q=the integer part of $\frac{a}{m}$ (the part to the left of the decimal point) and the remainder $r = m\frac{r}{m}$ is m times the part to the right of the decimal point. Or: repeatedly subtract/add multiples of m to a until you get a number between 0 and $m-1$; that is r, and then $a - r$ is a multiple of m, and q can be recovered by dividing. (That is: find multiples of m so that $a - qm = r$ is between 0 and $m-1$, then r must be the remainder and q must be the quotient!)

a and b are congruent mod m ($a \equiv b \pmod{m}$) if both have the same remainder on division by m; that is, $a - b$ is a multiple of m (notation: $m | a - b$, m divides $a - b$). The idea: a number is really the "same" as its remainder (the number line is "wrapped around" a circle going from 0 to $m-1$).
Basic facts: if $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$ then $a \equiv c \pmod{m}$ (i.e., <u>all</u> three have the same remainder mod m). If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $a + c \equiv b + d \pmod{m}$, $a - c \equiv b - d \pmod{m}$, and $a \cdot c \equiv b \cdot d \pmod{m}$ (the remainder of the sum is the sum of the remainders, etc.).

In the language of modular arithmetic, some popular check digit systems:

A basic sum check system: digits $a_1 a_2 \ldots a_k$, with a_k=check, chosen so that
$a_1 + a_2 + \cdots + a_k \equiv 0 \pmod{10}$.
UPC: digits $a_1 a_2 \ldots a_{11} a_{12}$, with a_{12}=check, chosen so that
$3a_1 + a_2 + 3a_3 + \cdots + 3a_{11} + a_{12} \equiv 0 \pmod{10}$. [groceries]
ISBN-10: digits $a_1 a_2 \ldots a_9 a_{10}$, with a_{10}=check=$0, \ldots, 9, X$ ($X = 10$), chosen so that
$10a_1 + 9a_2 + 8a_3 + \cdots + 2a_9 + a_{10} \equiv 0 \pmod{11}$. [books]
LUHN: digits $a_1 a_2 \ldots a_{15} a_{16}$, with a_{16}=check, chosen so that
$b_1 + a_2 + b_3 + \cdots + b_{15} + a_{16} \equiv 0 \pmod{10}$, <u>where</u> $b_i = 2a_i$ if $2a_i < 10$, otherwise $b_i = 2a_i - 9$. [credit cards]
mod 9 check: digits $a_1 a_2 \ldots a_k$, with a_k=check $=0, \ldots, 8$, chosen so that

the k-digit number $a_1a_2\dots a_k \equiv 0 \pmod 9$. [euro notes, Visa traveler's checks]
mod 7 check: digits $a_1a_2\dots a_k$, with a_k=check $=0,\dots,6$, chosen so that the k-digit number $a_1a_2\dots a_k \equiv 0 \pmod 7$. [UPS tracking, airline tickets]
mod m check: digits $a_1a_2\dots a_k$, with a_k=check $=0,\dots,m-1$, chosen so that the k-digit number $a_1a_2\dots a_k \equiv 0 \pmod m$.

Finding the check digit: call the check digit x and compute the appropriate sum; typically we end up solving $a+x$ =multiple of m, by finding the remainder r of a mod m and solving $r+x=m$.

Finding a missing/obliterated digit amounts to giving the unknown digit a name, x, and computing the sum; we end up solving $cx+a\equiv 0 \pmod m$. **Basic trick:** find d (if we can!) so that $dc\equiv 1 \pmod m$; then $0\equiv d(cx+a)\equiv (dc)x+(da)\equiv (1)x+(da)\equiv x+(da) \pmod m$, and solve as above! **Or,** by "brute force": plug each number from 0 to $m-1$ in for x in $cx+a$ to find all of the x which gives a multiple of m. Finding the d in the first approach can be done the same way; compute all of the $dc-1$ for $d=0,\dots,m-1$ until you find one that is a multiple of m.

For example, for UPC, can use $d=7$: $7\cdot 3=21\equiv 1 \pmod{10}$. For ISBN-10, every number $1,\dots,10$ has a corresponding number (e.g., to recover a_6 solve $5a_6+a\equiv 0 \pmod{11}$, and $9\cdot 5=45=44+1\equiv 1 \pmod{11}$, so $a_6+9a\equiv 9\cdot 5a_6+9a\equiv 0 \pmod{11}$).

Being able to recover a missing digit means we can detect changes in that digit's position: if there is only one answer, then any <u>other</u> answer would not yield something $\equiv 0$, <u>unless</u> we change the check digit! If more than one answer will work, then the system cannot detect the change of one answer to the other; the check digit remains the same (E.g., change 0 to 9 in the mod 9 system.)
We can test a system to see if it can detect transposition errors, by subtracting the two equations for the checks. For example, with UPC, transposing the first two digits cannot be detected if
$3a_1+a_2+3a_3+\cdots+3a_{11}+a_{12}\equiv 0 \pmod{10}$ and $3a_2+a_1+3a_3+\cdots+3a_{11}+a_{12}\equiv 0 \pmod{10}$. Subtracting, we get $2a_1-2a_2\equiv 0 \pmod{10}$, which requires $a_1-a_2=$ multiple of 5. So, e.g., UPC cannot detect the transposition of a 2 and a 7...

Simplifying the computation of a mod 9 check digit: $10\equiv 1 \pmod 9$, so $100=10\cdot 10\equiv 1\cdot 1=1 \pmod 9$, and so on, so $a_1a_2\dots a_k=a_1\cdot(10)^{k-1}+\cdots a_{k-1}\cdot 10+a_k\equiv a_1+\cdots+a_k$. Since we can always throw out multiples of 9 in these computations, we can throw out digits that add up to 9 (casting out 9's).

Simplifying the computation of a mod 7 check digit: $1\equiv 1 \pmod 7$, $10\equiv 3 \pmod 7$, $100=10\cdot 10\equiv 3\cdot 3=9\equiv 2 \pmod 7$, $1000=100\cdot 10\equiv 2\cdot 3=6 \pmod 7$, and so on [the pattern, we can work out, is $1,3,2,6,4,5,1,3,2,6,4,5,\dots$], so
$a_1a_2\dots a_k=a_1\cdot(10)^{k-1}+\cdots a_{k-1}\cdot 10+a_k\equiv a_k+3a_{k-1}+2a_{k-2}+6a_{k-3}+\cdots \pmod 7$.
A similar list of numbers can be created for any modulus.