Identification Numbers and Check Digits[1]

Many products or documents today have an identification number stamped. This numbers codes information about the product. Some examples:

1. Credit Cards
2. Postal service (ZIP and Bar Codes)
3. Grocery items (UPC)
4. Money orders
5. Dollar bills
6. Books (ISBN)
7. Driver licenses
8. Vehicle Identification Numbers

A Check Digit is a decimal (or alphanumeric) digit added to a number for the purpose of detecting the sorts of errors humans typically make on data entry.
It is not uncommon to make errors when handling numbers; like writing down a phone number, giving out a credit card number, writing down a street address, and so on. Even machines, like magnetic readers, are prone to errors. Some of the typical errors are of the following three types:

1. A single digit error For example, writing a 1 instead of a 7 somewhere in the number. This is the most common type of error, accounting for 60% to 90% of all errors.

¶ 1. Suppose that you have certain amount of money in your bank account. Lets say that you have a quantity between \$1 and \$1000. If you make a single digit error when balancing your checkbook, what is the greatest possible difference between your total and your bank's total? What is the least possible difference?

[1]Supported by NSF Grant 0502258

2. A transposition error where you reverse the order of two adjacent digits: for example, writing "51" instead of "15" somewhere in the number.

¶ **2.** Why is this true? If you make a transposition error when writing down a number, then the difference between this number and the original number is a multiple of 9.

3. A jump transposition error in which two non-adjacent digits are switched, for example you write 537 instead of 735.

¶ **3.** Can you devise a divisibility rule for detecting a jump transposition error?

A more detailed analysis and categorization of the sort of errors incurred when dealing with numbers was reported by J. Verhoeff (Error Detecting Decimal Codes, Mathematical Centre Tract 29, The Mathematical Centre, Amsterdam, 1969) based on a study of 12000 errors:

1. single errors: a becomes b (60% to 95% of all errors)
2. omitting or adding a digit (10% to 20%)
3. adjacent transpositions: ab becomes ba (10% to 20%)
4. twin errors: aa becomes bb (0.5% to 1.5%)
5. jump transpositions: acb becomes bca (0.5% to 1.5%)
6. jump twin errors: aca becomes bcb (below 1%, lower for longer jumps)
7. phonetic errors: $a0$ becomes $1a$, since the two have similar pronunciations in some languages, e.g. thirty and thirteen (0.5% to 1.5%)

The Check Digit and Check Equation We can eliminate (or easily detect) the problem of omitting or adding digits by restricting the input field to a given number of digits if we are dealing with numbers which are fixed in format, such as credit card numbers, Social Insurance Numbers, local phone numbers, and student ID numbers.
A method (Check Digit Scheme) for eliminating some of the other type of errors is the following. To an identification number a new digit is appended (usually as the last on the right) so that all the digits satisfy a predetermined equation, the Check Equation.
Other errors are detected by calculating whether the check equation for a particular check digit scheme is true. The check digit is included in the equation so that it is protected against errors as well. If the equation is not true, an error is present; if it is true, there may or may not be an error.
A number of different check digit schemes for detecting decimal number errors have been suggested, and several are in common use. We examine some of these schemes in the following pages.

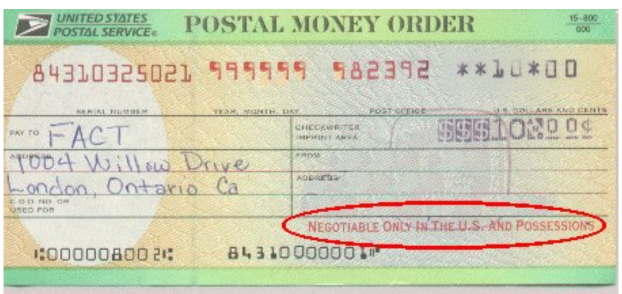
US Postal Money orders

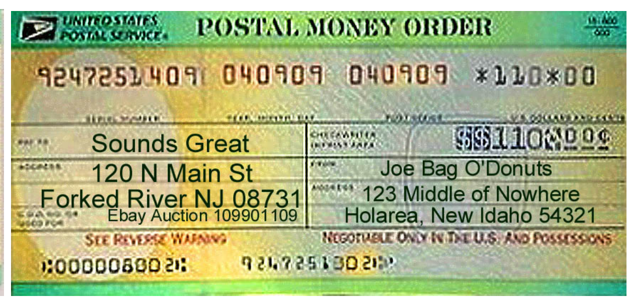
The US Post Office uses an identification number system for postal money orders. The identification number is 11 digits long, the last digit being the check digit. Thus, if the printed number on the money order is

$$a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}a_{11},$$

then

$$a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + a_9 + a_{10} = a_{11} \pmod 9$$





For example, the money order on the left has identification number 8431032502 and check digit 1. The sum

$$8 + 4 + 3 + 1 + 0 + 3 + 2 + 5 + 0 + 2 = 28$$

and $28 = 3 \times 9 + 1$ has remainder 1 after dividing by 9. The check digit is 1.

¶ **4.** The image of the money order on the right has been digitally altered and the check digit is missing. What is it?

¶ **5.** What type of errors does the US Postal Money order identification scheme detect? Suppose that a money order has the number `27120241168` and you erroneously copy that number as in the table below

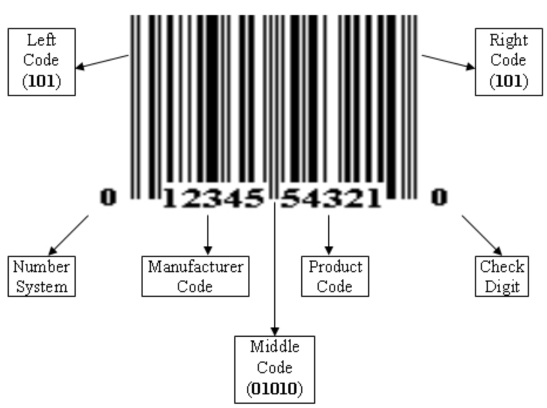
Copied Number	Error Type	Detected? (Yes or No)
`27120241168`		
`27129241168`		
`27123241168`		
`21720241168`		
`27120241186`		

UPC (Universal Product Code)

Grocery products use the so called UPC system for identifying products. A UPC number consists of 12 digits:

$$a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}a_{11}a_{12}.$$

The structure of the UPC is described in this picture



The first digit a_1 identifies the type of product, the next five digits $a_2a_3a_4a_5a_6$ identify the manufacturer, and the next five digits $a_7a_8a_9a_{10}a_{11}$ identify the product. The last digit a_{12} is the check digit, and it must satisfy:

$$3 \cdot a_1 + a_2 + 3 \cdot a_3 + a_4 + 3 \cdot a_5 + a_6 + 3 \cdot a_7 + a_8 + 3 \cdot a_9 + a_{10} + 3 \cdot a_{11} + a_{12} \equiv 0 \pmod{10}.$$

Digit a_1	Type of Product
0	General Groceries
2	Meat and Produce
3	Drugs and Health products
4	Non-food items
5	Coupons
6, 7	Other items

¶ 6. (a) Determine the check digit (?) for the UPC number `05074115017?`

(b) A box of corn-flakes has printed the following UPC code: `058000001277`. Is this correct?

(c) The UPC code `034?81923147` is missing a number (in the ?-position). What is this number?

¶ **7.** The "mod 9" scheme of the US Postal Money orders was not able to detect all of the single-digit errors, but the "mod 10" UPC scheme does. Suppose that the following UPC `036000291452`

0 36000 29145 2

is scanned resulting in the single digit errors below:

(a) `038000291452`

(b) `036000291453`

Which of the errors is detected?

ISBN International Standard Book Number

The International Standard Book Number (ISBN) uses a weighted code: Each digit is weighted according to its position in the number and the check digit is chosen so the weighted sum is evenly divisible by a prime number.
An ISBN number consists of 10 digits $a_1a_2a_3a_4a_5a_6a_7a_8a_9a_{10}$ so that the rightmost digit a_{10} satisfies the equation

$$10 \cdot a_1 + 9 \cdot a_2 + 8 \cdot a_3 + 7 \cdot a_4 + 6 \cdot a_5 + 5 \cdot a_6 + 4 \cdot a_7 + 3 \cdot a_8 + 2 \cdot a_9 + 1 \cdot a_{10} \cong 0 \pmod{11}$$

If the check digit a_{11} is 10, then the letter X is used instead.

The ISBN 10 digit number consists of four parts. These parts may be of differen lengths, and are usually separated by hyphens.

1. the group identifier (language-sharing country group),
2. the publisher code,
3. the item (title) number, and
4. a check digit.

The group identifier is a 1 to 5 digit number. The single first digit group identifiers are: 0 or 1 for English-speaking countries; 2 for French-speaking countries; 3 for German-speaking countries; 4 for Japanese; 5 for Russian; 7 for Chinese; and 8 for Spanish.

For example, the ISBN for my book *Foliations I* is `0-8218-0809-5` and the check equation is

$$10\times0+9\times8+8\times2+7\times1+6\times8+5\times0+4\times8+3\times0+2\times9+1\times5 \pmod{11} = 198 \pmod{11} = 0$$

This scheme detects any single error and the transposition of any two digits at any distance (assuming the overall number is 10 or fewer digits long).

¶ **8.** Determine the check digit a_{10} for the ISBN `0-669-33907?`

¶ **9.** Using the ISBN check digit scheme, determine which of the following are valid ISBN numbers

(a) `3-824-27519-X`

(b) `84-206-3613-4`

(c) `2729-60284-0`

¶ **10.** The ISBN `0-669-03925-4` is the result of the transposition of two adjacent digits not involving the first or last digit. Determine the correct ISBN.

¶ **11.** Prove that the ISBN code detects error arising from transposing the 1st and 3rd digits.

IBM Check Digit Scheme

The IBM Check Digit scheme is used to verifty the validity of credit cars numbers, and other ID numbers. It was created by IBM scientist Hans Peter Luhn, so it is also referred to as "Luhn algorithm."
This is a weighted code with parity dependent weights. The digits in the even positions (numbering from the right) are multiplied by 2, then reduced to a single digit (if > 9) by "casting out nines" (subtracting 9, which is equivalent to adding the digits). All digits are then summed and a check digit added to make the result evenly divisible by 10.
For example, given the number `5482 0923 1553 0413` the leading 5 is doubled, giving 10, which is then reduced to 1 by adding the digits of 10 together; similarly, the 8 becomes 16 and then 7; the 0 is oblivious to doubling; the 2 becomes 4; the 1 becomes 2; and the 5 in the second to last position becomes 10 and thus 1. Thus the check equation is

$$(5\#2+4+8\#2+2+0\#2+9+2\#2+3+1\#2+5+5\#2+3+0\#2+4+1\#2+3=50=0 \pmod{10}$$

where '#' represents multiplication "modulo 9;" for example $6\#2=12=3 \pmod 9$.
This scheme detects all single digit errors. It also detects most adjacent transpositions errors, but not those where 09 becomes 90. It also does not detect jump transposition errors, such as 553 becoming 355.

¶ **12.** The credit card number `5482 0923 1553 0413` is valid. Determine which of the following errors below are detected by the Luhn algortihm detect and which ones are not detected.

Copied Number	Error Type	Detected? (Yes or No)
`5482 0923 2553 0413`		
`5482 0923 1453 0413`		
`5482 9023 1553 0413`		
`5482 0921 3553 0413`		
`5482 0923 1355 0413`		

TI-84 Programs

The first program will compute the check digit for the ISBN check digit scheme. The program requests 9 digits and outputs a sible digit or the letter X

```
PROGRAM:ISBN
:Input N
:For(J,1,9)
:int(N/10^(9-J))►L1(J)
:N-L1(J)*10^(9-J)►N
:End
:0►N
:For(J,1,9)
:N+L1(J)*(11-J)►N
:End
:N-int(N/11)*11►N
:If N=0
:Disp 0
:If N=1
:Disp ‘‘X’’
:If N>1
:Disp 11-N
```

The next program implements the Luhn algortihm or IBM Check Digit scheme. Unfortunately, the TI-84 cannot handle 16 digit integers, so it cannot be used to validate credit card numbers, because these usually have 16 digits. It is possible to modify it so that it accepts 16 digit numbers, but that will obscure the nature of the algortihm.
The first part of the program (up to line 6) computes the length of the number `N`. Zeros on the left are ignored.

```
PROGRAM:LUHN
:Input N
:0►K
:While int(N/10^k)>0
:K+1►K
:End
:For(J,1,K)
:int(N/10^(k-J))►L1(K+1-J)
:N-int(N/10^(K-J))*10^(K-J)►N
:End
:For(J,2,K,2)
:If L1(J)>4
:2*L1(J)-9►L1(J)
:2*L1(J)►L1(J)
:End
:For(J,2,K)
:L1(1)+L1(J)►L1(1)
```

```
:End
:If L1(1)-int(L1(1)/10)*10=0
:Disp ‘‘NUMBER OK’’
:Disp ‘‘INVALID NO’’
```