**Project 2: Luhn Algorithm for Digit Validation**

**Project Objective:**

The objective of this project is to expose students to the Luhn Algorithm as a check-digit validation method. Intended for students in the first year programming course, this project integrates concepts in credit card numbers and IMEI numbers validation methods with string manipulation, text files, and use of regular expressions in this medium size programming assignment.

Target audience: CS2 students

**Luhn Algorithm for Digit Validation Project**

The purpose of this project is to learn more about text processing in an application using the Luhn algorithm for check-digit validation.

Luhn Algorithm

* The Luhn algorithm is also known as the “modulus 10” or “mod 10” algorithm
* Created by IBM scientist Hans Peter Luhn, patent filed in 1954 and was granted in 1960
* It is a checksum algorithm to validate identification numbers and is used in validating credit card numbers, IMEI numbers (International mobile equipment identifications)
* Designed to protect against accidental errors, not intended to be a cryptographically secure hash function or to protect against malicious attacks

Example:

Assume an account number is 7 9 9 2 7 3 9 8 7 1 x (where x is the check digit)

Starting from the check digit, move left, starting with the first digit, double every other digit to yield

 7 18 9 4 7 6 9 16 7 2 -

For every number that became two-digits after doubling,

 7 18 9 4 7 6 9 16 7 2 -

to yield ..

 7 9 9 4 7 6 9 7 7 2 -

Now add all these numbers together, sum of digits is 67.

Multiply this sum of digits by 9. 67 x 9 = 603

Take the product 603, mod 10. 603 % 10 = 3

* Check digit is 3

Strengths and weaknesses

Strengths – algorithm is easy to implement and can work on strings of numbers of any length (can pad strings on the left with insignificant zeroes)

Weakness – can detect single-digit errors but cannot detect transposing of digits

Luhn algorithm is often used by credit card companies to generate the final digit of a credit card.

A typical credit card number is composed of 4 sets of 4-digit numbers. For example,

Check digit

Account number

First 6 digits are Bank ID (BIN) or Issuer ID (IIN)

1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8

The first 6 digits are Bank ID (BIN) or Issuer ID (IIN).

The second group of 9-digits is the account number.

The last digit is the check digit.

Credit Card Types BIN Prefix Credit Card number length

American Express 34 | 37 15

Discover 6011 | 622126 – 622925 | 644 – 649 | 65 16

MasterCard 51 – 55 16

Visa 4 13 – 16

Example:

Consider a credit card number, 4 8 0 1 5 2 1 5 7 3 9 9 7 4 0 8

Since the card number begins with 4, it is a Visa card. The first 6 digits are Bank ID / Issuer ID.

Take the account number, 1 5 7 3 9 9 7 4 0

From the right, double every other digit:

 2 5 14 3 18 9 14 4 0

Add double digits,

 2 5 5 3 9 9 5 4 0

Sum of digits is 42.

Multiply 42 by 9 gives 378.

378 mod 10 gives 8, which is the check digit on the number on the card.

Variations on the Luhn algorithm

Please be advised that there are variations on the Luhn algorithm. One variation on computing the check digit is to use the amount needed to reach a number divisible by 10. For example, if the sum of digits is 84, the check digit would be 6 since adding 6 to 84 brings us to 90, which is the next number divisible by 10.

Programming Project Description

The objective of this project is to accept input from a text file. Each line of the text file is a sequence of digits with no spaces between digits, representing a credit card number.

Your program will read in each line of text from the file and determine the credit card type by checking the BIN. It is possible that the data file contains card numbers from a bank / issuer other than the 4 major credit card companies. Any other card types other than the 4 card companies described above are not acceptable in your application. A message of “Unknown card type” should be displayed, and the card number will be rejected.

Use the Luhn algorithm as described in this guide to determine whether the credit card number is valid. The output of your program for each credit card number should be the credit card type, the credit card number, and whether the card is acceptable. If the card is not acceptable, your output will also state whether the reason is a non-acceptable BIN or an invalid card number.

References:

<http://searchsecurity.techtarget.com/definition/LUHN-formula>

<http://umairj.com/374/how-to-test-credit-card-numbers-using-luhns-algorithm/>

<https://en.wikipedia.org/wiki/Luhn_algorithm>