CSC 111aD – Fall 2013 Due 5pm Friday, 10/18/13

Lab 8

You may not be aware that credit card numbers are not just made up of random digits. There's a method to their selection. They contain several pieces of information that can be used for performing fast validity tests. For example, Visa card numbers always begin with 4, and any valid Visa card number must pass a digit-sum test known as the Luhn checksum algorithm. Luhn's algorithm states that if you sum the digits of a valid Visa number in a certain way, the total sum will be a multiple of 10. Systems that accept credit cards perform a Luhn test before contacting the credit card company to make sure the number is active. This lets the merchant block fake or mistyped credit card numbers before contacting the credit card company. As an aside, Canadian Social Insurance Numbers must pass this same test.

The algorithm for summing the digits is as follows. Consider each digit of the credit card to have a numbered position: the leftmost digit is at position 0, and the rightmost is at position 15. Process each digit one at a time. For digits at odd-numbered indexes, simply add that digit to a running sum. For digits at even-numbered indexes, double the digit's value, then if that doubled value is less than 10, add it to the sum; on the other hand, if the doubled number is 10 or greater, add each of the digits separately into the sum.

The following algorithm describes Luhn's algorithm:

```
sum = 0.
for each digit of credit card number,
    if the digit's index is odd,     # counting left to right
        add the digit to sum.
    else,
        double the digit's value.
        if the doubled value is less than 10,
            add the doubled value to sum.
        else,
            split the doubled value into its two digits.
            add the first digit to sum.
            add the second digit to sum.
if the sum is a multiple of 10 return True (Valid), else return False (Invalid).
```

4111-1111-1111-1111 and 4408-0412-5436-9873 are example credit card numbers that pass the Luhn algorithm. The following figure shows the algorithm summing the second number in detail, after the hyphens were removed. Notice how digits at odd indexes are doubled and potentially split into two digits if they exceed 10 when doubled. For example, the number 7 at index 7 is doubled to 14 then split to make 1+4.

An example checksum computation using the Luhn algorithm.

```
CC #    4408-0412-5436-9873

        4   4   0   8   0   4   1   2    7    4   3   6    9    8    5    3
Scale  *2      *2      *2      *2       *2       *2       *2        *2
       ---------------------------------------------------------------------
        8   4   0   8   0   4   2   2   14    4   6   6   18    8   10    3

Sum   = 8 + 4 + 0 + 8 + 0 + 4 + 2 + 2 +(1+4)+ 4 + 6 + 6 +(1+8)+ 8 +(1+0)+ 3
      = 70
```

70 is divisible by 10, therefore this card number is valid, according to this test.

CSC 111aD – Fall 2013 Due 5pm Friday, 10/18/13

Lab 8

Step 1 –
Download Lab_8.py from the Resource→Labs link on Sakai. You should not need to modify the main function.

Step 2 –
Complete the function named stripHyphens(). The purpose is to take a string as a parameter and return an equivalent string with all hyphens removed. Test it to make sure it works correctly.

Step 3 –
Complete the function named `checkVisa()` using Luhn's algorithm described previously. The function should return True if the credit card number passed as an argument checks out okay, else it should return False. Debug and test your program with both valid and invalid credit card numbers.

Step 4 –
Upload your completed assignment to Sakai by 5pm Friday, 10/18/2013.

Scoring:
6 pts – stripHyphens() works correctly to remove hyphens from any string
3 pts – stripHyphens() is well documented
8 pts – checkVisa() works correctly to return True/False as appropriate
3 pts – checkVisa() is well documented