CSC6010  **Programming Assignment #1 Fall’17**

**Financial: Credit card number validation.**

**Credit card numbers follows certain patterns. A credit card number must have between 13 and 16 digits. It must start with:**

* + **4 for Visa cards**
  + **5 for Master cards**
  + **37 for American Express cards**
  + **6 for Discover cards**

**In 1954, *Habs Luhn* of IBM proposed an algorithm for validating credit card numbers. The algorithm is useful to determine whether a card number is entered correctly or whether a credit card is scanned correctly by a scanner. Credit card numbers are generated following this validity check, commonly known as the *Luhn check or Mod 10 check*, which is described as follows ( for illustration, consider the card number 4388576018402626):**

1. **Double every second digit from right to left. If double of a digit results in a two-digit number, add up the two digits to get a single digit number.**
2. **Now add all single-digit number from the step above (Step 1 ).**

**4 + 4 + 8 + 2 + 3 + 1 + 7 + 8 = 37**

1. **Add all digits in the odd places from right to left in the card number.**
   1. **6 + 6 + 0 + 8 + 0 + 7 + 8 + 3 = 38**
2. **Sum the results from the above two step ( Step 2 and Step 3).**

**37 + 38 = 75**

1. **If the result from the above step ( step 4 ) is divisible by 10, the card is valid; otherwise, the card is invalid. For example, the card number 4388576018402626 is invalid, but the card number 4388576018410707 is valid.**

**Design and Implement the above algorithm in Java *Recursively*. Your algorithm should prompt the user to enter a credit card number as a long integer. Display whether the number is a valid or invalid credit card number. Design your program to use the following methods: These methods should be done recursively… Make sure you have several runs, i.e. several testing conditions.**

**//Return if the card number is valid**

**Public static Boolean isValid(long number)**

**//Get the result from step 2**

**Public static int sumOfDoubleEvenPlace( long number )**

**//Return this number it is a single digit, otherwise, return the sum of the two //digits**

**Public static int getDigit( int number )**

**//Return sum of sumOfOddPlace( long number )**

**Public static int sumOfOddPlace( long number )**

**//Return the number of digits in digit**

**Public static int getSize( long number )**

**//Return the first k number of digits from the number. If the number of digits in number is less than k, return number**

**Public static long getPrefix( long number, int k )**

**Here are some runs of the program**

**Enter a credit card number as a long integer**

**4388576018410707**

**4388576018410707 is a valid credit card number.**

**Enter a credit card number as a long integer**

**4388576018402626**

**4388576018402626 is a valid credit card number.**