1.

**CPE考試日期：**2017/9/26

**CPE題目編號：**11743

**CPE題目名稱**：Credit Check

2.

**題目：**

These days, it has become commonplace to make purchases over the internet using a credit card. However, because credit card numbers are relatively long, it is easy to make a mistake while typing them in. In order to quickly identify errors like typos, most e-commerce websites use a checksum algorithm to verify credit card numbers. One popular checksum algorithm is the Luhn algorithm, which can detect any single-digit error as well as many common multiple-digit errors: 1. Starting with the second-last digit and moving backwards, double every other digit to obtain a list of numbers. 2. Add up the digits of these numbers, then add the undoubled digits from the original number. Sum the two results. 3. If the total ends in a 0, the credit card number is valid, and it is invalid otherwise. For example, using the number 5181 2710 9900 0012: 1. Double the appropriate digits (5181 2710 9900 0012) to obtain the values: 10, 16, 4, 2, 18, 0, 0, 2. 2. Add up the digits of these values to get (1+0) + (1+6) + 4 + 2 + (1+8) + 0 + 0 + 2 = 25. The sum of the undoubled digits is 1+1+7+0+9+0+0+2 = 20, so the total is 20+25=45. 3. 45 does not end in a 0, so this credit card number is invalid. For this problem, you must write a program that checks the validity of credit card numbers according to the Luhn algorithm.

**中文題目說明：**

這些日子以來，使用信用卡在網路上購買東西已經變的司空見慣。

但是因為信用卡卡號比較長，很容易在輸入他們的時候打錯。

為了快速的識別錯誤，如數字打錯，大多數的電子商務網站都會用一種校檢演算法來確認信用卡卡號

一種較為流行的校檢演算法叫做 "Luhn"演算法 (Luhn algorithm)，它可以檢測任何一位元的錯誤及多位元錯誤：

1.從倒數第二個位元開始，將他們放到後面，並且加倍其他沒有移動的位元到另一個列表

2.把列表內的數字的位元加總(n)，再把被移到後面的數字加總(m)，在把兩個數加起來 (n+m)

3.如果這個數字的結尾是0，則信用卡卡號為合法的，反之則是不合法的。

這裡有個例子，以這組號碼為例 5181 2710 9900 0012:

1.把相對應的數字加倍後，放到另一個列表 (**5** 1**8**1 **2**7**1**0  **9**9**0**0  **0**0**1**2) :10，16，4，2，18，0，0，2。

2.把這些數的位元加起來得到 (1+0) + (1+6) + 4 + 2 + (1+8) + 0 + 0 + 2=25

  沒有對應到的位元合為1+1+7+0+9+0+0+2 = 20 ， 所以最後的總和是 20 + 25 = 45。

3. 45不是以0結尾，故這組信用卡號並不合法。

對於這個問題，你需要寫一個根據 Luhn演算法的程式來確認輸入的信用卡號是否合法。

**※輸入資料說明**

輸入的第一行為N 代表接下來有幾組測試資料

之後的N行

每行包含一行信用卡號碼

每個信用卡號碼由16個 10進位位數字組成四組,並且以空白分隔

**※輸出資料說明**

輸出包含一行文字

如果號碼是合法的,請輸出"Valid"

如果不合法,請輸出"Invalid"

3.**執行範例**

* 執行畫面範例題型a-單筆 (綠色為輸入，直接剪自eclipse的console)

|  |
| --- |
| ? 5181 2710 9900 0012  Invalid |
| ? 5181 2710 9900 0017  Valid |

* 執行畫面範例題型b-多筆 (綠色為輸入，直接剪自eclipse的console)

|  |
| --- |
| ? 2  5181 2710 9900 0012  5181 2710 9900 0017  Invalid  Valid |

* 執行畫面範例題型c-多筆 (綠色為輸入，直接剪自eclipse的console)

|  |
| --- |
| ? 2  5181 2710 9900 0012  5181 2710 9900 0017  -1  Invalid  Valid |

4.**參考解答**

* 單筆輸入

|  |
| --- |
| **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String args[]) {  Scanner sc = **new** Scanner(System.***in***);  System.***out***.print("? ");  **int** sum1 = 0, sum2 = 0;  **for** (**int** i = 1; i <= 4; i++) {  **int** num = sc.nextInt();  sum2 += num % 10;  num /= 10;  **int** a1 = 2 \* (num % 10);  sum1 += (a1 >= 10) ? (a1 % 10 + a1 / 10) : a1;  num /= 10;  sum2 += num % 10;  num /= 10;  a1 = 2 \* (num % 10);  sum1 += (a1 >= 10) ? (a1 % 10 + a1 / 10) : a1;  }  **if** ((sum1 + sum2) % 10 == 0)  System.***out***.println("Valid");  **else**  System.***out***.println("Invalid");  }  } |