Assignment 1: Check Digit Computation

Due: 20:00, Tue 27 Sep 2016 Full marks: 100

Introduction

Every credit card number contains a check digit at its rightmost, which is used for simple error detection. It can be used to protect against accidental errors such as a mistyped digit or the permutation of two successive digits. In this assignment, you will write a program that computes the check digit of a partial credit card number. For simplicity, we assume that a card number has exactly 16 digits in the form $d_1d_2d_3d_4d_5d_6d_7d_8d_9d_{10}d_{11}d_{12}d_{13}d_{14}d_{15}d_{16}$, in which $d_1 \ldots d_{15}$ is the 15-digit partial card number obtained from the program user and d_{16} is the check digit to be computed.

To compute the check digit of a partial card number, we can use the *Luhn algorithm*[a] described below:

1. Double the odd-positioned digits d_1, d_3, d_5, d_7, d_9, d_{11}, d_{13}, and d_{15}.
2. If a doubled digit is greater than 9, replace it by its sum of digits. (E.g., 16 is replaced by 1 + 6 = 7.)
3. Sum the even-positioned digits d_2, d_4, d_6, d_8, d_{10}, d_{12}, and d_{14} with the modified odd-positioned digits d_1, d_3, d_5, d_7, d_9, d_{11}, d_{13}, and d_{15}.
4. Multiply the sum by 9. Then the units digit (個位數, the rightmost digit) of the multiplication is the check digit d_{16}.

Example 1: partial card number is 763545841927506.

1. Double the underlined odd-positioned digits in $\underline{7}6\underline{3}5\underline{4}5\underline{8}4\underline{1}9\underline{2}7\underline{5}0\underline{6}$: $(\underline{7} \times 2) = 14$, $(3 \times 2) = 6$, $(4 \times 2) = 8$, $(\underline{8} \times 2) = 16$, $(\underline{1} \times 2) = 2$, $(\underline{2} \times 2) = 4$, $(\underline{5} \times 2) = 10$, and $(\underline{6} \times 2) = 12$.
2. Replace 14, 16, 10 and 12 by their sum of digits: 14 → 1 + 4 = 5, 16 → 1 + 6 = 7, 10 → 1 + 0 = 1, and 12 → 1 + 2 = 3.
3. Sum all 15 digits: $5 + 6 + 6 + 5 + 8 + 5 + 7 + 4 + 2 + 9 + 4 + 7 + 1 + 0 + 3 = 72$. (The digits in red came from steps 1 and 2.)
4. Multiply the sum 72 by 9: $72 \times 9 = 648$. The units digit (8) is the check digit. Therefore, the full card number is: 7635458419275068.

Example 2: partial card number is 543210987654321.

1. Double the underlined digits in $5\underline{4}3\underline{2}1\underline{0}9\underline{8}7\underline{6}5\underline{4}3\underline{2}1$: $(\underline{5} \times 2) = 10$, $(3 \times 2) = 6$, $(1 \times 2) = 2$, $(9 \times 2) = 18$, $(\underline{7} \times 2) = 14$, $(\underline{5} \times 2) = 10$, $(\underline{3} \times 2) = 6$, and $(\underline{1} \times 2) = 2$.
2. Replace 10, 18, 14 and 10 by their sum of digits: 10 → 1 + 0 = 1, 18 → 1 + 8 = 9, 14 → 1 + 4 = 5, and 10 → 1 + 0 = 1.
3. Sum all 15 digits: $1 + 4 + 6 + 2 + 2 + 0 + 9 + 8 + 5 + 6 + 1 + 4 + 6 + 2 + 2 = 58$. (The digits in red came from steps 1 and 2.)
4. Multiply the sum 58 by 9: $58 \times 9 = 522$. The units digit (2) is the check digit. Therefore, the full card number is: 5432109876543212.

[a] Reference: http://en.wikipedia.org/wiki/Luhn_algorithm

Program Specification

The program should *repeatedly* accept a user input until a negative integer is entered. You can assume that *the user input is always either a 15-digit number or a negative integer*. (That is, you do not have to check whether the input is out of this assumption.) If the input is a 15-digit number, then you have to use the Luhn algorithm to compute the check digit and display the full card number. The full card number should be printed as four pieces of 4-digit segments in the format XXXX-XXXX-XXXX-XXXX. If the input is a negative integer, then the program terminates.

Note that the `int` data type is not big enough to represent 15-digit numbers. In order to store the partial or full card numbers, you can declare variables of the `long long` type, which is a bigger integral data type.

```
long long num;    // int num; does not work
```

You are *not allowed to use arrays* in this assignment.

Program Output

The following shows some sample output of the program. The **bold blue** text is user input and the other text is the program output. You can try the provided sample program for other input. *Your program output should be exactly the same as the sample program* (i.e., same text, same symbols, same letter case, same number of spaces, etc.). Otherwise, it will be considered as *wrong*, even if you have computed the correct result.

```
Enter a 15-digit partial card num: 763545841927506↲
Full card num is: 7635-4584-1927-5068
Enter a 15-digit partial card num: 543210987654321↲
Full card num is: 5432-1098-7654-3212
Enter a 15-digit partial card num: 432100080012056↲
Full card num is: 4321-0008-0012-0564
Enter a 15-digit partial card num: -1↲
Bye!
```

Submission and Marking

- Your program file name should be `luhn.cpp`. Submit the file in Blackboard (https://elearn.cuhk.edu.hk/).
- Insert your name, student ID, and e-mail address as comments at the beginning of your source file.
- You can submit your assignment multiple times. Only the latest submission counts.
- Your program should be *free of compilation errors and warnings*.
- Your program should *include suitable comments as documentation*.
- *Plagiarism* is strictly monitored and *heavily punished* if proven. Lending your work to others is subjected to the same penalty as the copier.