College of Computer Studies

DEPARTMENT OF
SOFTWARE TECHNOLOGY

Hands-On Exam 2: DLSU Student Debit Card Concept

Due to their simplicity of use and practical pay-back choices, credit and debit cards have become a necessary part of our life. Such cards offer discounts, offers, and deals that are unsurpassed by any other financial goods and are a gold mine for the savvy user.

The front of a credit/debit card has essential parts, namely: the chip, the card number, expiration date, and the cardholder's name.

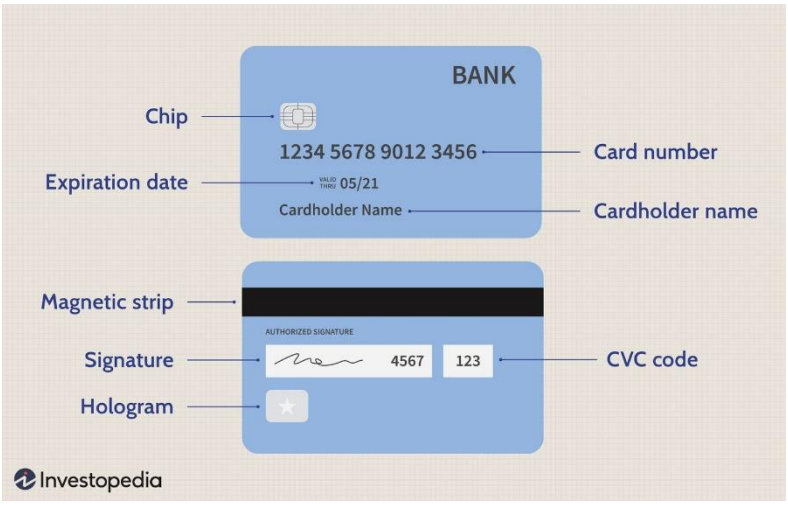

Figure 1 – Parts of a Credit Card
(from https://www.investopedia.com/terms/c/creditcard.asp)

The card number has distinctive qualities:
1. It usually consists of **12 or 16 digits** (depending on the bank).
2. Moreover, these digits are not just random numbers. These are actually sequenced and computed based on the Luhn's Algorithm.

The Luhn method, commonly referred to as the modulus 10 or mod 10 algorithm, is a straight-forward checksum technique used to validate a multitude of identifying numbers, including IMEI numbers and credit card numbers.

Most credit cards and many government identification numbers use the algorithm as a simple method of distinguishing valid numbers from mistyped or otherwise incorrect numbers. It was designed to protect against accidental errors, not malicious attacks.

For this scenario, the school has decided to improve the use of the ID cards of students, faculty, and staff to include a corresponding debit account that everyone can load and use for cashless transactions. (Smells like an interesting project for future courses… hmmm?)

How Luhn's Algorithm Work

Let's assume we want to validate the number: 4417 1234 5678 9113.

1. Starting from the rightmost number, double every other digit.

Card Digit	4	4	1	7	1	2	3	4	5	6	7	8	9	1	1	3
Multiplier	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
	8	4	2	7	2	2	6	4	10	6	14	8	18	1	2	3

2. If the result of the doubling ends up with two digits, then add those two digits together:

Card Digit	4	4	1	7	1	2	3	4	5	6	7	8	9	1	1	3
Multiplier	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
	8	4	2	7	2	2	6	4	10	6	14	8	18	1	2	3

	8	4	2	7	2	2	6	4	1+0	6	1+4	8	1+8	1	2	3
	8	4	2	7	2	2	6	4	1	6	5	8	9	1	2	3

3. Add up all numbers

	8	4	2	7	2	2	6	4	1	6	5	8	9	1	2	3
Sum	70															

4. If the final sum is divisible by 10, then the card number is valid. If it is not divisible by 10, the number is invalid or fake.

PROGRAM OVERVIEW:

For this hands-on exam, you will be tasked to create a program that will ask the user to enter a sixteen-digit number (inputted as two 8-digit numbers*). Once the input is provided, the program will display whether or not the card number is valid or invalid.

***NOTE**: The maximum value for an integer in C is 2147483647 (10 digits). However, what is required to us as input in the program is a 16-digit number. Therefore, there is a need to enter 2 integers (maximum of 8 digits each) to suffice the required input. The first integer input will represent the first 8 digits of the card number and the second integer will represent the last 8 digits. For example, if the 1st input number is 44171234 and if the 2nd input number is 56789113, then the card number is 4417123456789113

TASKS TO DO:

You are required to complete the codes for the following functions in LASTNAME-CCAlgo.c:

`int isNumberSameDigitLength(int nCardNumber, int nTotalDigits)` [10 points]
This function checks if the number of digits of nCardNumber is the same as value nTotalDigits.

Example:
- **`isNumberSameDigitLength(12345678, 5)`** will return **0** (i.e., We are checking if the first parameter has 5 digits)
- **`isNumberSameDigitLength(1234, 4)`** will return **1** (i.e., Since 1234 has 4 digits)

`int extractDigit(int nCardNumber, int nPosition)` [15 points]
This extracts the nth digit of nCardNumber. (NOTE: We start counting digits from the RIGHT)

Example:
- **`extractDigit(12345678, 5)`** will return **4** (i.e., Extract the 5th digit starting from the right)
- **`extractDigit(12345678, 8)`** will return **1** (i.e., Extract the 8th digit starting from the right)

`int isValidCardNumber(int nCardNumberP1, int nCardNumberP2)` [25 points]
The function validates if the card number provided by the user is valid or not. You to integrate the functions **`extractDigit`** and **`computeDigitSum`** for this function. Please refer to the mechanics on the Luhn's Algorithm in page 2 of the specifications.

Example:
- **`isValidCardNumber(51001089, 33421116)`** returns 1. (i.e., the number is a valid card number)
- **`isValidCardNumber(55001293, 61823442)`** returns 0. (i.e., the number is NOT a valid card number)

You are given the following files for this problem:
- LASTNAME-CCAlgo.c -- skeleton file which contains some initial code that you'll need to complete.
- DebitCredit.c -- contains the main() function for testing purposes; make sure to read this file as to see how the program should execute.

DELIVERABLES:

Submit this **LASTNAME-CCAlgo.c** and replace the LASTNAME with your own. For example, if your last name is **ANG**, you must upload your files as **ANG-CCAlgo.c**.

TESTING & SCORING:
- Your program will be compiled via gcc -Wall. Thus, for each function that does not compile successfully, the score for that function is 0.
- Your program will be tested by your instructor with other main() (which may contain different values from the ones given to you) and with function calls of different parameter values.

- Full credit will be given for the function only if the student's implementation is correct for all the test values used by the instructor during checking AND only if the student's implementation complied with the requirement and did not violate restrictions. Deductions will be given if not all test cases produce correct results. No credit is given if restrictions were not followed.

TEST RUNS:

RUN 1:

```
CREDIT / DEBIT CARD NUMBER VERIFIER
===================================

When entering the card number, add a
space/gap between the 8th and 9th
digit.
Enter a card number : 51001089 33421116

Card has a VALID number!
```

RUN 2:

```
CREDIT / DEBIT CARD NUMBER VERIFIER
===================================

When entering the card number, add a
space/gap between the 8th and 9th
digit.
Enter a card number : 59002021 18138603

Card has a VALID number!
```

RUN 3:

```
CREDIT / DEBIT CARD NUMBER VERIFIER
===================================

When entering the card number, add a
space/gap between the 8th and 9th
digit.
Enter a card number : 55001293 61823442

Card has an INVALID number!
```

RUN 4:

```
CREDIT / DEBIT CARD NUMBER VERIFIER
===================================

When entering the card number, add a
space/gap between the 8th and 9th
digit.
Enter a card number : 12345678 5678
Please recheck your input.

CREDIT / DEBIT CARD NUMBER VERIFIER
===================================

When entering the card number, add a
space/gap between the 8th and 9th
digit.
Enter a card number : 00123456 98765432
Please recheck your input.

CREDIT / DEBIT CARD NUMBER VERIFIER
===================================

When entering the card number, add a
space/gap between the 8th and 9th
digit.
Enter a card number : 56001224 62318622
Card has a VALID number!
```

//grs