Singapore Institute of Management

UOW AUSTRALIA

ASSIGNMENT 2 – PART 1

CSIT110– Programming Fundamental Using Python

Session 3: July to September 2024

INSTRUCTIONS TO CANDIDATES

1. The assignment consists of two parts. This is part 1 of the assignment.
2. Part 2 is Moodle quiz. Should be done in class.
3. The name of the program must be **YourName_A2.**py (Only one Python file); remember to replace **YourName** by your actual "shorter" name.
4. **Total mark of Assignment 2 is 6 marks; 4 marks for Part II.**

Your program, should begin with

# Full Name:
# Tutorial Group
# Declaration: …… tell me if it is your own work …. and whether you have
# passed your program to your friends.

Objective

The following features should be explored in your design:

- Selection and repetition designs
- The use of user defined functions

Don't forget to include this statement in your program

input ("Press enter to terminate")

Task 6 marks

Online payments are becoming more and more popular. They usually ask customers to enter their credit card numbers and bypass by a checking system and verify if the credit cards entered are valid. Most of the banks use the same algorithm for verification.

The LUHN formula (don’t worry about the name given to this formula), also known as the modulus 10 algorithm, was created in the late 1960s. Shortly thereafter, credit card companies started using it. Because the algorithm is in the public domain, it can be used by anyone. Currently it's used by credit card companies, libraries, pharmacies, many motor vehicle divisions etc. Almost any institution that needs a unique account or identification numbers uses the LUHN algorithm.

A few credit card companies use 16 digits separated by 4 tokens displayed on the credit card (my visa and master card show that☺).

Let us assume that we have the following credit card number issued by a bank: (You can assume that user enter the credit card numbers separated by hyphens)

`2323-2005-7766-3554`

For each of the tokens, you reverse the numbers (we remove the hyphen too)

`3232 5002 6677 4553`

For every token, you multiply by 2 every digit in the *even* position, i.e. positions 2 and 4 (left to right for each token). We have the following new four tokens:

$$\mathtt{3\underline{4}3\underline{4}\ \ 5\underline{0}0\underline{4}\ \ 6\underline{12}7\underline{14}\ \ 4\underline{10}5\underline{6}}$$

You then find the sum of digits of each token:

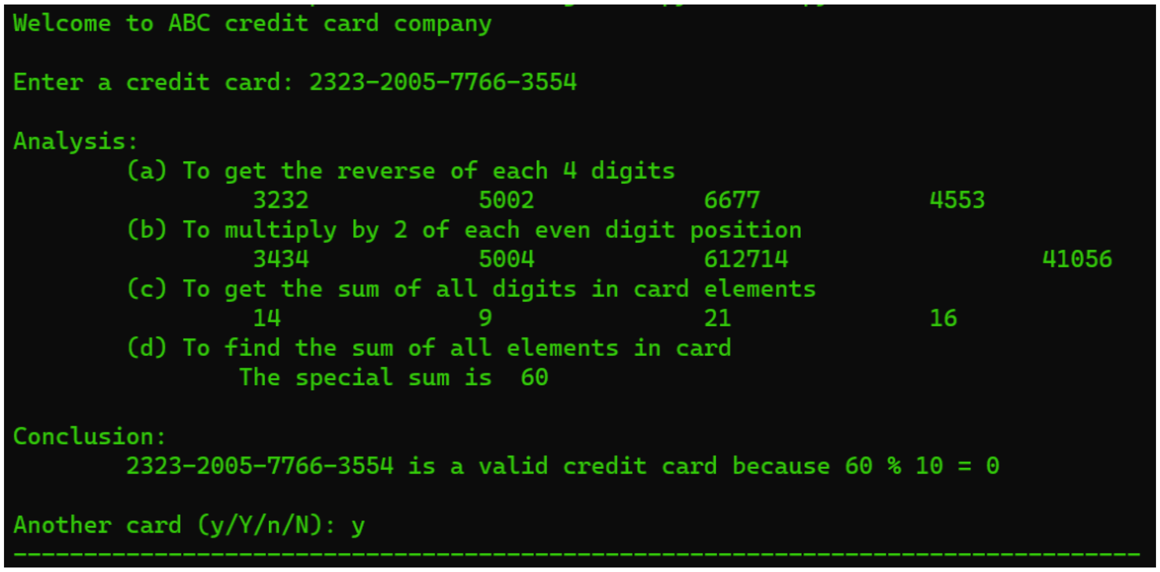
`14   9   21  16`

Finally add the 4 numbers together

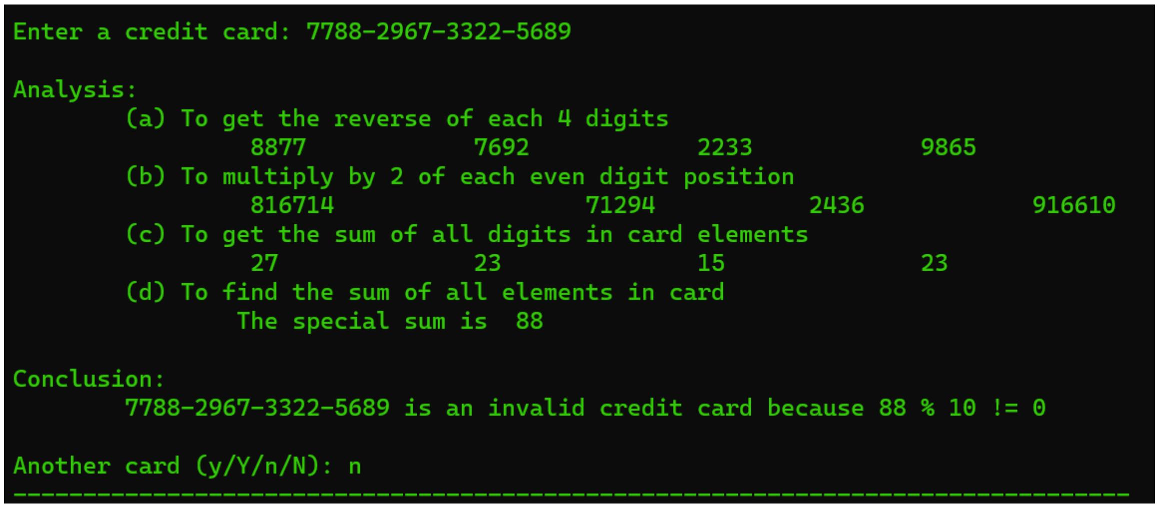
`14 + 9 + 21 + 16 = 60`

Divide the total by **`10`**, if the remainder is **`0`** then the number is valid. In our case, **`60 % 10 = 0`**, therefore the credit card number **`2323-2005-7766-3554`** is valid.

After knowing this algorithm, we wish to design a program to check whether a credit card issued by a company is valid. Upon executing your program, it shows the following interactions and displays:



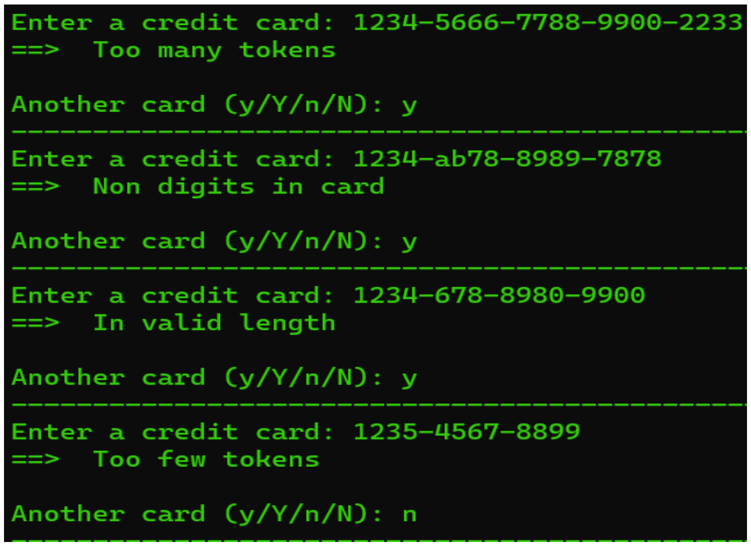
```
Welcome to ABC credit card company

Enter a credit card: 2323-2005-7766-3554

Analysis:
        (a) To get the reverse of each 4 digits
                3232            5002            6677            4553
        (b) To multiply by 2 of each even digit position
                3434            5004            612714                  41056
        (c) To get the sum of all digits in card elements
                14              9               21              16
        (d) To find the sum of all elements in card
               The special sum is  60

Conclusion:
        2323-2005-7766-3554 is a valid credit card because 60 % 10 = 0

Another card (y/Y/n/N): y
--------------------------------------------------------------------------------
```


```
Enter a credit card: 7788-2967-3322-5689

Analysis:
        (a) To get the reverse of each 4 digits
                 8877            7692            2233            9865
        (b) To multiply by 2 of each even digit position
                 816714                  71294           2436            916610
        (c) To get the sum of all digits in card elements
                 27              23              15              23
        (d) To find the sum of all elements in card
                The special sum is  88

Conclusion:
        7788-2967-3322-5689 is an invalid credit card because 88 % 10 != 0

Another card (y/Y/n/N): n
--------------------------------------------------------------------------------
```

If you don't get the same results as shown in the above screen, do not panic as I may have some logical errors in the design☺. However, I used my very **old** visa and master cards issued by a local bank to test my system, and they worked!!! Probably these banks use "my" algorithm. Sorry I won't be able to give you my new card numbers as I will use them to crash your system☺.

In the design, you certainly need a few important user-defined functions:

- A function to find the sum of digits of a given positive integer.
- A function to get the reverse of a positive integer.
- A function to form a new integer so that the 2nd the 4th digits are doubles.
- A function to check if a credit card is valid: A few suggestions … feel free to add in more if you think of others ….


```
Enter a credit card: 1234-5666-7788-9900-2233
==>  Too many tokens

Another card (y/Y/n/N): y
---------------------------------------------
Enter a credit card: 1234-ab78-8989-7878
==>  Non digits in card

Another card (y/Y/n/N): y
---------------------------------------------
Enter a credit card: 1234-678-8980-9900
==>  In valid length

Another card (y/Y/n/N): y
---------------------------------------------
Enter a credit card: 1235-4567-8899
==>  Too few tokens

Another card (y/Y/n/N): n
---------------------------------------------
```

- Some other useful functions required by your design.

IMPORTANT

The name of your program must be **YourName_A2.py** and make sure that this file can be executed. Upload **ONLY** the Python file to Moodle**. ALL ZIP FILE SUBMISSION WILL BE REJECTED**

No re-submission will be allowed after grading.

In the above file, remember to put down your name and the following declaration (some similar contents):

# Tell me if it is your own work, and whether you have passed your

# program to your friends etc etc etc

# and willing to accept whatever penalty given to you.

- **Wrong file name -0.2 mark**
- **No declaration, no name etc -0.3 mark**
- **No demo -0.5**