CSE101

DUE APR. 12

23:59 KST

ASSIGNMENT II

PROFESSOR

FRANCOIS

RAMEAU

LIST & LOOPS

General instructions

- Please add your name and email in the dedicated location at the top of your code
- Do not use any external library, except when explicitly requested
- Try to follow the naming convention proposed by PEP-8 seen in class
- Use ONLY what we have already seen in class. If not, you will get ZERO points
- Use meaningful names for your variables and functions
- If you face problem submitting your code via GitHub please contact the professor and the TA by email
- Note that the received code will be tested on a classifier to detect potential usage of Large Language Model. We will also pay particular attention to plagiarism
- Leave comments in your code to explain your code and describe the difficulties you faced

INVITATION LINK

https://classroom.github.com/a/2-CHNHUF

Exercise 1: Compute statistic on a list (3 points)

For this first exercise, you will perform a few statistical analyses on a vector of length n (expressed as a list in python): $v = [v_1 \quad v_2 \quad \cdots \quad v_n]$. You should complete the function `listStatistic` such that it returns the following information:

- The **maximum** value in the vector
- The **minimum** value of the vector
- The **average** of the vector $\bar{v} = \frac{\sum_{i=1}^{n} v_i}{n}$

You should use a single FOR loop to resolve this problem. The results should be returned as a tuple containing `max, min, average` (in this order).

After finalizing this function, you will write another function `listStd` to compute the **standard deviation** σ of the vector v. For reference, the equation for the standard deviation is the following:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(v_i - \bar{v})^2}{n-1}}$$

Take advantage of your previous function `listStatistic` to compute the average needed for this calculation.

Recap

1.	Compute the maximum of the vector	0.5 point
2.	Compute the minimum of the vector	0.5 point
3.	Compute the average of the vector	1 point
4.	Compute the standard deviation of the vector	1 point

. Exercise 2: The Lunh algorithm (7 points)

The Luhn algorithm is a simple checksum formula used to validate identification numbers, such as credit card numbers. Using this very simple algorithm, we will determine if a credit card number is fake or not!

We will develop the Lunh algorithm in successive small steps, which will make you manipulate lists using FOR loops.

1. Write the credit card number (1 point)

Let's consider the following credit card:

In the `main` function of the assignment, you will declare a list called `card_nb` which will contain this credit card number (each digit will be one element of the list, so you should end up with a vector of length 16)

2. Reverse the list (1 points)

You will now fill the function `reverseList`, the goal of this function is to reverse the order of the input list. Thus, this function will take the variable `card_nb` as an input and return its reversed version that you might call `reversed_card_nb`.

For example, here is the expected behavior of this function:

`card_nb`	4	0	0	0	0	0	1	2	3	4	5	6	7	8	9	9
`reversed_card_nb`	9	9	8	7	6	5	4	3	2	1	0	0	0	0	0	4

3. Double the value of one element over two (1 point)

Now you will take this reversed card number and multiply each element having an odd index by a factor two. You will implement this capability in the function `doubleOddIndex`.

Given our reversed card number, the expected output that we can call `lunh_product` is depicted in the following table where the odd index elements are highlighted in blue

`reversed_card_nb`	9	9	8	7	6	5	4	3	2	1	0	0	0	0	0	4
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Factor	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
`lunh_product`	9	18	8	14	6	10	4	6	2	2	0	0	0	0	0	8

4. Deal with double digits element (1 point)

You will know edit the function `digitSum`. This function receives a single integer as input. If this integer is composed of two digits it will sum their values such that it returns a single digit output.

Here are a few examples:

- if the input integer is 18, it consists of 2 digits so our function will return 1+8=9.
- if the input integer is 4, it consists of a single digit so our function will return 4.
- if the input integer is 12, it consists of 2 digits so our function will return 1+2=3.
- if the input integer is 9, it consists of a single digit so our function will return 9.

5. Replace every double digit of `lunh_product` by a single digit using `digitSum` (1 point)

You will now edit the function `replaceDoubleDigit`. This function will iterate through the `lunh_product` list and replace every double digits using our previous function `digitSum`.

The following table illustrates the behavior of our function `replaceDoubleDigit`, every double digits element are highlighted in blue:

`lunh_product`	9	18	8	14	6	10	4	6	2	2	0	0	0	0	0	8
explanation		1+8		1+4		1+0										
`single_digit_prod`	9	9	8	5	6	1	4	6	2	2	0	0	0	0	0	8

6. Check the validity of the card (1 point)

Finally, in the function `isCardValid` you will sum all the elements of `single_digit_prod` and if the last digit of this sum is a "0" then the card is valid. If it is any other digit then the card is invalid.

For instance, if the resulting sum is "70" the card is valid because the last digit of 70 is a zero. Imagine if the output is 89, the last digit here is a 9 which means that the card is invalid

7. Put everything together (1 point)

Let's now, put everything together in the function `lunhAlgorithm`:

1. Reverse the `card_nb` list
2. Multiply every element on odd indexes
3. Replace every double-digit numbers by summing their individual digit values
4. Check the validity of the card

Your function `lunhAlgorithm` should return a Boolean True if the card is valid and False if it is not.

Now can you tell me if the card in this example is valid or not by running algorithm?