Goals
By the end of this assignment, you will have
  •  practice with Python lists
  •  more practice with Python loops
  •  more practice with docstrings, good programming style, and automated testing

(0) (based on GCM p. 145 #7) Complete the examples in the docstring and then write the body of the following function:

```python
def same_first_last(L):
    """(list) -> boolean

    Precondition: len(L) >= 2

    Returns True if the first item of the list is the same as the last; else returns False.
    >>> same_first_last([3, 4, 2, 8, 3])
    True
    >>> same_first_last(['apple', 'banana', 'pear'])
    >>> same_first_last([4.0, 4.5])
    ""
```

(1) (based on GCM ch. 8 #8) Complete the examples in the docstring and then write the body of the following function:

```python
def is_longer(L1, L2):
    """(list, list) -> boolean

    Return True if the length of L1 is longer than the length of L2; else return False
    >>> is_longer([1, 2, 3], [4, 5])
    True
    >>> is_longer(['abcdef'], ['ab', 'cd'])
    >>> is_longer(['a', 'b', 'c'], [1, 2, 3])
    ""
```

(2) Write a function, frequent, with one parameter, psw, a string. If psw is in a list of frequently used passwords (password, One234!, qwerty, letmein, trustno1, 111111, passw0rd), frequent should return False; otherwise, return True.
(3) Write a function, `mySum`, with one parameter, a list of numbers. `mySum` returns the sum of the numbers in the input list. (`mySum` should not use Python's built-in `sum` function.) For example,

```python
>>> mySum([1, 2, 3])
6
```

(b) Test `mySum` with examples of calls to `mySum` with simple, normal, and borderline data. Include these in `mySum`'s docstring and test `mySum` using `testmod` from module `doctest`.

(4) (a) Write a function `middle(L)` which takes a list `L` as its argument, and returns the item in the middle position of `L`, when `L` has an odd length. Otherwise, `middle` should return 999999. For example, calling `middle([8, 0, 100, 12, 1])` should return 100. Hint: remember the `//` integer division operator. Hint: consider writing an auxiliary function, `odd`.

(b) Test `middle` with examples of calls to `middle` with simple, normal, and borderline data (testing all paths through the function). Include these in `middle`'s docstring and test `middle` using the `testmod` function from the `doctest` module.

(5) (CS Circles) You have been hired to write a function which checks if a given credit card number is valid. Your function `check(S)` should take a string `S` as input. First, if the string does not follow the format "#### #### #### ####", then it should return False. Then, if any # is not a digit, then it should return False. Then, if the sum of the digits is divisible by 10, then the procedure should return True, else it should return False. For example, if `S` is the string "9384 3495 3297 0123" then although the format is correct, the digit sum is 72 so you should return False. Zero is not considered to be divisible by 10, that is, a string of all zeros should return False. Hint: investigate string methods `isdigit` and `split`.

Function `check` should work for the following test cases:

```python
>>> check('2768 3424 2345 2358')
False
>>> check('9384 3495 3297 0121')
True
>>> check('1876 0954 325009182')
False
>>> check('0000000000000000')
False
>>> check('0000 0000 0000 000')
False
>>> check('0 0 0 0000000000000000')
False
>>> check('')
False
>>> check('0000 0000')
False
>>> check('0123 4567 8902 4568')
True
>>> check('0123 4567 89AB EFGH')
False
>>> check('0123 4567 89AB 5555')
False
```
(Challenge) Update the grade_calculator function so that it has two parameters, exams and projects, each of which is a list of lists of scores and adjustment factors. Note that the parameter lists mean that the user will no longer be asked to enter scores. For example:

```python
grade_calculator(
    [[25, 1.17],[38, .875],[50, .583],[35, .875],[60, .5]],
    [[39, .4],[40, .5]])
81 B
```

**Grading Rubric**

This project will be marked out of 70 points. Each function (same_first_last, is_longer, frequent, mySum, middle, check) is worth 10 points: 2 points for the function header, 3 points for the docstring, 2 points for the body of the function, 1 point for the return statement, and 2 points for correct output. my_sum and middle test cases are 3 points each. For function check, the body of the function is worth an additional 4 points (6 total).

**Getting Started**

**Write your program using an editor window**, so you can save your file.

Programming style is important! Remember
- Include a docstring in every function
- Use whitespace between operators and operands
- Use descriptive variable names
- Add appropriate comments.

**Finishing & submitting your work**

When you have completed all of the problems, add additional comments to your code to make sure the functions for each problem are clear. Check the list of functions above to make sure that your Python file includes all of them (same_first_last, is_longer, frequent, mySum, middle, and check). Do a final Save command to save the code in the editor window as a file with the name cis122project8.

To submit your project, login to Blackboard. From the menu on the left hand side of the screen, choose "Projects". In the projects folder, choose "Submit Loops and Lists". In Section 2 of the page that is displayed, scroll down to "Attach file" and choose "Browse My Computer". Locate cis122project8.py (the file you just created), and double click on it. The file name will appear on the "Attached Files" list.

At the bottom of Section 2, you will see a "Comments" window. **This is where you credit all of the sources of any help you may have received on this assignment, including your partner if you are working in a programming pair.** This is also the place to include any feedback you may have about the assignment and/or any remaining questions you may have.

Scroll down to Section 3 and hit the "Submit" button. You may re-submit your project up until the project deadline. Only the final submission will be graded.