Week 6
In-Class Mini Trial Lab Test (30 minutes)

General Instructions:

1. You will perform the lab test on your personal laptop. You will need to join the WebEx session on your laptop, and the Zoom session using your **mobile phone** before the test starts.

2. You are not allowed to communicate with anyone or access any network during the test.

4. You may refer to any offline content on your laptop during the test.

5. Make sure your code can generate exactly the same output as we show in the sample runs. You may be penalized for missing spaces, missing punctuation marks, misspelling, etc. in the output.

6. Do not hardcode. We will use different test cases from the provided ones to test and grade your solutions.

7. Follow standard Python coding conventions (e.g. naming functions and variables).

8. Python script file that cannot be executed will NOT be marked and hence you will be awarded 0 marks. You may wish to comment out the parts in your code which cause execution errors.

9. Include your name as author in the comments of all your submitted source files. For example, include the following block of comments at the beginning of each source file you need to submit.

```
# Name: Lum Ting Wong
# Email ID: lum.ting.wong.2019
```

Instructions on how to submit your solutions:

1. Before the test begins, you will be instructed to download the resource files of the test from eLearn.

2. You will be given a password to unzip the downloaded .zip file. After unzipping the file, **you should rename the folder to your email id.** For example, if your SMU email is lum.ting.wong.2019@sis.smu.edu.sg, you should rename the folder to lum.ting.wong.2019. You need to save your solutions to this folder for submission later. You may be penalized for not following our instructions.

3. When the test ends, you will be instructed to submit your solutions as a single zip file to eLearn.

Q1: Add First Odd Digits [**]

Define a function called `add_first_odd_digits` in `q1.py`. The function takes in a list of strings called `str_list` as its parameter. The function goes through all the strings in this list and for each string, it looks for a digit that is an odd number (i.e., 1, 3, 5, 7 or 9). The function adds up the first odd digit in each string and returns the sum. If a string does not contain any odd digit, then nothing is added for that string.

E.g., `add_first_odd_digits(['abc123def', 'SMU2345SIS', 'XYZ0', '7777'])` returns 11. This is because the first odd digit found in `'abc123def'` is 1, the first odd digit found in `'SMU2345SIS'` is 3, `'XYZ0'` doesn't contain any odd digit, and the first odd digit in `'7777'` is 7. Therefore, we get 1 + 3 + 7 = 11.

If `str_list` is empty, then 0 is returned by the function.

Use the provided **q1_test.py** to test your code. DO NOT modify **q1_test.py**.

Q2: Display Strings [**]

Define a function called `display_strings()` inside the file `q2.py`.

The function takes in the following parameters:

- a list of strings called `str_list`
- a character called `ch`

The function **prints** out the strings in `str_list`, one in each row. However, each string will have some spaces in front such that the output looks like a staircase. Moreover, strings that are shorter than the longest string in `str_list` will be "padded" with `ch` in the end so that they reach the same length as the longest string.

For example, if `str_list` is `['Spider Man', 'Iron Man', 'Hulk', 'Thor', 'Captain America', 'Black Widow']`, and if `ch` is `'-'`, then the following output should be displayed:

```
     Spider Man-----
    Iron Man-------
   Hulk-----------
  Thor-----------
 Captain America
Black Widow----
```

We can see that the longest string in `str_list` is `'Captain America'`, which has 13 characters. Therefore, each string is padded with some `'-'` such that it becomes 13-character long.

If `str_list` is `['ABC', 'B', 'CD', 'D']`, and `ch` is '*', then the output is

```
   ABC
  B**
 CD*
D**
```

If `str_list` is an empty list, then nothing needs to be printed.

Note:

- The last row of the output has no space in front.
- You can use `print(ch*n)` to print out the character `ch` for `n` times. E.g., `print('@'*3)` will print `'@@@'`.

Use the provided **q2_test.py** to test your code. DO NOT modify **q2_test.py**.