Lab 4 - Lists

Q1: Code Tracing [**]

What's the output of the following code?

```
def modify_list(my_list):
    for index in range(len(my_list)):
        x = my_list[index]
        if len(x) > 5:
            my_list[index] = x[0:5]

str_list = ["IS111", "Python", "Programming", "List"]

modify_list(str_list)
print(str_list)
```

What's the output of the following code?

```
def modify_list(my_list):
    for element in my_list:
        if len(element) > 5:
            element = element[0:5]

str_list = ["IS111", "Python", "Programming", "List"]

modify_list(str_list)
print(str_list)
```

Q2: Email Extractor [**]

Part I

Implement a function called `extract_email_id()` that extracts email ID from an email address. The function takes in a string as a parameter. This string is an email address. The function **returns** the front part of the email address before the '@' symbol.

For example,

- `extract_email_id("jerry.lee@sis.smu.edu.sg")` returns the string `"jerry.lee"`.
- `extract_email_id("alan_wong@gmail.com")` returns the string `"alan_wong"`.
- `extract_email_id("alan_wong.com")` returns `""`.

If the given string does not contain the '@' symbol, the function returns an empty string.

Part II

Implement another function called `extract_multiple_email_ids()`. This function takes in a string that contains multiple email addresses, separated by semi-colons. (You can assume that semi-colons cannot be used as part of an email address.) The function **prints out** the email IDs one by one in separate rows. The function doesn't return anything.

For example, calling `extract_multiple_email_ids("jerry.lee@sis.smu.edu.sg;alan_wong@gmail.com;george_tan@yahoo.com")` gives the following output:

```
jerry.lee
alan_wong
george_tan
```

Hint: You can use the `split()` method on a string.

Q3: Check Username [***]

A website allows people to sign up as members, but when choosing their usernames, there are the following restrictions:

- The username cannot be empty nor contain any space.
- Each character in the username must be a lowercase letter, a digit, or one of the following special symbols: `_.!#$%?`
- The length of the username cannot exceed 8.

Write a function that checks whether a string is a valid username. The function takes in a string as its parameter and **returns** `True` or `False`.

Try the following test cases, which should yield the following results:

Argument	Returned value
`"abcdefgh"`	True
`"abcdefghi"`	False
`"ab$cd"`	True
`"ab_cd"`	True
`"ab-cd"`	False
`"ab:cd"`	False
`""`	False
`"ab cd"`	False
`"abcDef"`	False
`'abc8ef"`	True

Q4: List of Strings

a) [**] Implement a function called `get_avg_len()` that takes in a list of strings. The function returns the **average length** of all the strings in the given list.

For example, `get_avg_len(["C", "Java", "Python", "PHP"])` returns `3.5`.

If the list is empty or if the strings are all empty strings, the function returns 0.

b) [**] Implement a function called `get_longest_str()` that takes in a list of strings. The function returns the string that is the **longest** among all the strings in the given list. If there are more multiple strings having the longest length, the function returns the **first** such string in the list.

For example, `get_longest_str(["C", "Java", "Python", "PHP"])` returns `"Python"`, and `get_longest_str(["C", "Java", "HTML", "PHP"])` returns `"Java"`.

If the list is empty, the function returns an empty string.

c) [***] Implement a function called `concatenate_emails()`. This function takes in a list of strings, where some of these strings are email address. The function **returns** a string that contains the email addresses separated by semi-colons.

Here a string is considered an email address if it contains exactly one '@' symbol and it does not contain any space.

For example, `concatenate_emails(["IS111", "a @ b", "jerry.lee@sis.smu.edu.sg", "@@@", "alan_wong@gmail.com", "Python", "george_tan@yahoo.com"])` returns `"jerry.lee@sis.smu.edu.sg;alan_wong@gmail.com;george_tan@yahoo.com"`.

If the list is empty, the function returns an empty string.

d) [***] Implement a function called `check_hashtags()`. The function takes in a list of strings. The function returns `True` if all the strings are hashtags, i.e., all the strings start with a '#' symbol and do not contain any space. The function **returns** `False` if at least one of the strings is not a hashtag.

For example,

- `check_hashtags(["#singapore", "#music", "#travel"])` returns `True`
- `check_hashtags(["#singapore", "#music album", "#travel"])` returns `False`

- `check_hashtags(["singapore", "#music", "#travel"])` returns `False`

If the list is empty, the function returns `False`.

Use the test code provided in the notebook to test your function. It should provide the following output:

```
True
False
False
False
```