Trial Lab Test 2 [45 minutes, in-class]

Q1 [**]

In `q1.py`, implement a function called `get_ppl_with_fever()`. This function takes in the following parameter:

- `ppl_list`: This is a list of tuples, where each tuple consists of two elements:
  - The first element is a person's name (of `str` type).
  - The second element is a *non-empty* list of numbers (where each number is of `float` type). Each number represents a temperature reading of this person.

The function returns a list that contains the names of people whose average temperature (over all his/her temperature readings) is above 37.5 (i.e., avg_temperature > 37.5).

If `ppl_list` is empty, the function returns an empty list.

See the following examples.

- Example #1: `get_ppl_with_fever([('John', [37.6, 37.8]), ('George', [38.1, 37.2, 37.5]), ('Vivian', [36.7])])` returns `['John', 'George']`. This is because John's average temperature is (37.6+37.8)/2=37.7, George's average temperature is (38.1 + 37.2 + 37.5)/3= 37.6, but Vivian's average temperature is 36.7. So only John and George's average temperatures are above 37.5 but Vivian's is not.

- Example #2: `get_ppl_with_fever([])` returns `[]`.

- Example #3: `get_ppl_with_fever([('Gideon', [36.5, 36.7]), ('Mary', [37.6, 36.8, 36.9, 36.8])])` returns `[]` because nobody's average temperature is above 37.5.

Run `q1_test.py` to test your code.

Q2 [**]:

You are given a file that contains multiple lines. Each line shows several groups of integer numbers. The numbers are separated by single spaces, and the groups are separated by the symbol `'*'`. For example, the file may look like the following:

```
10 20 30*100 400 200 300*-10 -20*9 3
1000**15 45 60
```

Note that some groups have only one number, and some groups have zero number. For example, the second row above contains three groups of numbers, where the first group has only a single number 1000, the second group is empty (between the two *s), and the third group has 3 numbers: 15, 45 and 60.

In `q2.py`, define a function called `process_numbers()` that takes in two parameters:

- The name of an input file as described above.
- The name of an output file.

The function should process the input file and write the following information to the output file:

- For each line of the input file, write the number of groups in that line to the output file.
- For each group of numbers in each line of the input file, find the maximum number in the group and write the maximum number to the output file. Use `*` to separate the maximum number of different groups. If a group has zero number inside (i.e., an empty group), write `NA`.
- Each line of the input file has a corresponding line in the output file showing the information above.

For example, given the file shown above, the output file should look like the following:

```
4: 30*400*-10*9
3: 1000*NA*60
```

Run `q2_test.py` to test your code.

Q3 [***]

In the file `q3.py`, define a function called `create_email_dict()`. The function takes in a parameter called `email_list`, which is a list of strings where each string is a valid SMU email address. You can assume that there are not duplicate email addresses in the list.

Specifically, each email address in the list consists of an email ID and a domain, separated by `@`.

- Each email ID consists of lowercase letters, dots (.) and optionally digits. There're two types of email IDs:
    - Student email ID: These email IDs always end with `".yyyy"`, where `"yyyy"` are 4 digits indicating the year in which the student is enrolled.
    - Staff email ID: These email IDs don't have any digit.
- Each domain is either `"smu.edu.sg"` or `"xxx.smu.edu.sg"` where `"xxx"` indicates one of the schools at SMU.

The function returns a dictionary where each key is a string consisting of a school name and a year (e.g., `"scis-2021"`) and the value for this key is a list of email addresses found in `email_list` that belong to a student in the corresponding school and enrolled in the corresponding year. Email addresses in `email_list` that are not student emails or do not contain a school name are ignored.

Examples:

- Given the following `email_list`:

```
['abc@smu.edu.sg', 'xyz.2021@scis.smu.edu.sg',
'john.smith.2020@soe.smu.edu.sg',
'alice.lin.2020@smu.edu.sg',
'dd.2021@scis.smu.edu.sg']
```

  the function returns the following dictionary:

```
{'scis-2021': ['xyz.2021@scis.smu.edu.sg',
               'dd.2021@scis.smu.edu.sg'],
 'soe-2020': ['john.smith.2020@soe.smu.edu.sg']}
```

- Given the following `email_list`:

```
['abc@smu.edu.sg', 'xyz.2021@smu.edu.sg',
'a.b.c@soa.smu.edu.sg']
```

  the function returns an empty dictionary.

Run `q3_test.py` to test your code.