Trial Lab Test 2 v1.0 [40 minutes, in-class]

Q1: [**]

Part (a)

Define a function called `find_last_letter()` that takes in a string as its parameter. The function returns the index of the ***last*** occurrence of a letter (either uppercase or lowercase) in the string, or -1 if no letter exists in the string.

E.g.,

- `find_last_letter('12abc345')` returns `4`.
- `find_last_letter('12ab345C')` returns `7`.
- `find_last_letter('12345')` returns `-1`.
- `find_last_letter('A')` returns `0`.
- `find_last_letter('')` returns `-1`.

Part (b)

Define a function called `find_second_letter()` that takes in a string as its parameter. The function returns the index of the ***second*** occurrence of a letter (either uppercase or lowercase) in the string, or -1 if no letter exists in the string or only one letter exists in the string.

E.g.,

- `find_second_letter('12abc345')` returns `3`.
- `find_second_letter('ABcd')` returns `1`.
- `find_second_letter('12345')` returns `-1`.
- `find_second_letter('A')` returns `-1`.
- `find_second_letter('')` returns `-1`.

Q2: [**]

Define a function called `get_reversed_list()` that takes in a list of numbers as its parameter. Call this list `num_list`. The function returns a list that ***reverses*** the original list and turns every number into its ***opposite*** number.

E.g.,

- `get_reversed_list([1.5, 2.5, 3.9])` returns `[-3.9, -2.5, -1.5]`.
- `get_reversed_list([4.5, 0.0])` returns `[0.0, -4.5]`.
- `get_reversed_list([])` returns `[]`.

Q3 [***]:

You are given a file called “numbers.txt” that contains multiple lines. Each line shows several groups of 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#100 300 200 400#-10 -20 0#1.0 3.0
1000 3000#15 45 60#5.0
```

In q3.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, calculate the average of the numbers in the group and write the average value to the output file. Use ‘#’ to separate the average values of different groups.
- Each line of the input file has a corresponding line in the output file showing the information above.
- In the end of the output file, write the total number of groups found in the input file and the maximum average value among all the groups.

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

```
4: 20.0#250.0#-10.0#2.0
3: 2000.0#40.0#5.0
Total number of groups: 7
Maximum average: 2000.0
```

You can assume that the input file contains at least one line, and each line of the input file contains at least one group, and each group contains at least one number.