CS1010S Programming Methodology
National University of Singapore

Practical Examination

12 Nov 2022

Time allowed: 1 hour 30 minutes

Instructions (please read carefully):

1. This is **an open-book exam**. You are allowed to bring in any course or reference materials in printed form. No electronic media or storage devices are allowed.
2. This practical exam consists of **three** questions (one question with two sub-questions). The time allowed for solving this test is **1 hour 30 minutes**.
3. The maximum score of this test is **20 marks**. Note that the number of marks awarded for each question **IS NOT** correlated with the difficulty of the question.
4. You are advised to attempt all questions. Even if you cannot solve a question correctly, you are likely to get some partial credit for a credible attempt.
5. While you are also provided with the template `practical-template.py` to work with, your answers should be submitted on Coursemology. Note that you can **only run the test cases on Coursemology for a limited number of tries** because they are only for checking that your code is submitted correctly. You are expected to test your own code for correctness using IDLE and not depend only on the provided test cases. Do ensure that you submit your answers correctly by running the test cases at least once.
6. In case there are problems with Coursemology.org and we are not able to upload the answers to Coursemology.org, you will be required to name your file `<mat no>.py` where `<mat no>` is your matriculation number and leave the file on the Desktop. If your file is not named correctly, we will choose any Python file found at random.
7. Please note that it shall be your responsibility to ensure that your solution is submitted correctly to Coursemology and correctly left in the desktop folder at the end of the examination. Failure to do so will render you liable to receiving a grade of **ZERO** for the Practical Exam, or the parts that are not uploaded correctly.
8. Please note that while sample executions are given, it is **not sufficient to write programs that simply satisfy the given examples**. Your programs will be tested on other inputs and they should exhibit the required behaviours as specified by the problems to get full credit. There is no need for you to submit test cases.
9. While you may use any built-in function provided by Python, you may not import functions from any packages, unless otherwise stated and allowed by the question.

GOOD LUCK!

Question 1 : Intervals [5 marks]

An interval is a pair of integers in the form `(a, b)` where $a < b$. It represents an interval of integers from a to b inclusive.

Two intervals can be merged to be represented by one interval if they overlap. The following helper function `merge` takes in two intervals as inputs, and returns a new merged interval if the two intervals overlap, or `False` otherwise. The implementation is given as follows:

```
def merge(a, b):
    if a[1] < b[0] or b[1] < a[0]:
        return False
    else:
        return (min(a[0], b[0]), max(a[1], b[1]))
```

Using `merge`, **your task** is to implement the function `merge_intervals` that takes in an arbitrary number of intervals and returns a list with the minimum number of merged intervals that covers all the intervals in the input. You can return the list of intervals in **any order**.

[5 marks]

Sample execution:

```
>>> merge_intervals((1, 2), (3, 4))
[(1, 2), (3, 4)]

>>> merge_intervals((1, 2), (2, 4))
[(1, 4)]

>>> merge_intervals((1, 2), (3, 4), (2, 6))
[(1, 6)]

>>> merge_intervals((7, 12), (3, 9), (1, 4))
[(1, 12)]

>>> merge_intervals((7, 9), (3, 5), (10, 13), (1, 15))
[(1, 15)]
```

Question 2 : YouTube Statistics [10 marks]

You own a YouTube channel and are interested in knowing different statistics related to YouTube videos and channels. You access the publicly available dataset containing the details of views and likes of trending videos on different channels for several months.

Important note: You are provided with a data file `YouTubeStat.csv` for this question for testing, but your code should work correctly for *any* data file with the same column format and *will* be tested on other data files. There is no inherent ordering among different rows in the data files. There might be multiple entries for a video on a channel that is trending for many months.

Each row in the data file represents a video id belonging to a channel, with the first row being the header. The data file contains the following columns in the following fixed format:

video id	channel title	views	likes

A. You first want know how your channels are faring. Write a function `get_video_id` that takes in a *channel title* and the *csv file* as input and returns the number of unique video IDs of the given channel that have more than 1K views and 1K likes.

[5 marks]

Sample Execution:

```
>>> get_video_id('NFL', 'YouTubeStat.csv')
13
>>> get_video_id('HBO', 'YouTubeStat.csv')
2
>>> get_video_id('20th Century Fox', 'YouTubeStat.csv')
4
```

B. You realize that videos on your channels are not very popular. You want to understand the kind of channels that usually trends and include similar videos to your channel. To this end, you want to determine the top-k trending channels based on the ratio of the total number of views and the total number of likes of all videos on the channel. Write a function `top_k_channels` that

1. calculates total views and total likes obtained by all videos for each channel title;
2. calculates the ratio of total views and total likes for each channel title, rounded to 1 decimal place;
3. returns the top-k channel titles along with the ratio.

Top-k here means that you should rank the channel titles according to their ratio and take the first k channel titles. If the remaining channels, from the $(k+1)$-th and so on, have the same ratio as the k-th channel, you should also include those channel titles as well.

Note that you can use in-built `round(num, places)` to perform rounding.

[5 marks]

Sample execution:

```
>>> top_k_channels(2, 'YouTubeStat.csv')
[('Made by Google', 2183792.0), ('MarenMorris', 939565.8)]
>>> top_k_channels(5, 'YouTubeStat.csv')
[('Made by Google', 2183792.0), ('MarenMorris', 939565.8), ('Douglas
   Thron', 929068.2), ('Tinder', 875988.2), ('Huckabee', 279484.8)]
>>> top_k_channels(7, 'YouTubeStat.csv')
[('Made by Google', 2183792.0), ('MarenMorris', 939565.8), ('Douglas
   Thron', 929068.2), ('Tinder', 875988.2), ('Huckabee', 279484.8),
   ('Jhené Aiko', 252895.5), ('Las Vegas Metropolitan Police',
   198139.4)]
>>> top_k_channels(8, 'YouTubeStat.csv')
[('Made by Google', 2183792.0), ('MarenMorris', 939565.8), ('Douglas
   Thron', 929068.2), ('Tinder', 875988.2), ('Huckabee', 279484.8),
   ('Jhené Aiko', 252895.5), ('Las Vegas Metropolitan Police',
   198139.4), ('Snapchat', 138256.2)]
```

Question 3 : COVID and its aftermath [5 marks]

Two years have passed and COVID has become a new normal for us. New variants of the disease appearing every now and then from mutations of existing variants. Contracting COVID gives immunity to the contracted variant and its predecessors.

In this question, we will model the interaction between `COVID` variants and `Person`.

A variant of `COVID` is initialised with a name. It has a method `mutate` which takes in a name and returns a new `COVID` variant which is mutated from itself.

A `Person` can be infected by a `COVID` variant. When infected, the person gains immunity to the variant, as well as all the variants it was mutated from. For example, if Bravo variant mutates to Lambda which mutates to Omicron, getting infected by Omicron would also give immunity to Bravo, Lambda as well.

A `Person` can be infected with multiple variants at the same time, and if they have not recovered from COVID, they will pass on their infected variants to other persons they meet with.

`Person` is initialised by their name and supports the following methods:

- `infect` takes a `COVID` variant as input, and returns a string based on the following conditions:
  - `'<person name> contracted <covid name>'` if the Person is not immune. The Person is infected with the variant and gains the required immunity.
  - `'<person name> is immune to <covid name>'` if the Person is immune to the variant.
- `recover` takes no inputs returns a string based on the following conditions:
  - `'<person name> recovers from COVID'` if the Person is currently infected. Person then recovers from all current infection.
  - `'<person name> is COVID negative'` if the Person is not currently infected with any variant.
- `meet` takes another `Person` as input. Any person who is currently infected will spread all infected variants to the other person. The method has no return value, i.e. `None`.

Provide an implementation for the classes `COVID` and `Person`.

NOTE. Pay close attention to the output strings. Even a single space or punctuation mark may lead to failure for a test case.

You are advised to solve this problem incrementally. Even if you cannot fulfil all the desired behaviours, you can still get partial credit for fulfilling the basic behaviours.

Sample execution is given as follows:

```
>>> alpha = COVID('Alpha')
>>> delta = alpha.mutate('Delta')

>>> bravo = COVID('Bravo')
>>> gamma = bravo.mutate('Gamma')
>>> lambd = bravo.mutate('Lambda')
```

```
>>> omicron = lambd.mutate('Omicron')

>>> waikay = Person('Wai Kay')
>>> ashish = Person('Ashish')
>>> nitya = Person('Nitya')

>>> waikay.infect(delta)
'Wai Kay contracted Delta'
>>> waikay.infect(alpha)
'Wai Kay is immune to Alpha'
>>> waikay.recover()
'Wai Kay recovers from COVID'
>>> waikay.infect(delta)
'Wai Kay is immune to Delta'

>>> waikay.meet(ashish)

>>> waikay.infect(lambd)
'Wai Kay contracted Lambda'
>>> waikay.infect(bravo)
'Wai Kay is immune to Bravo'
>>> waikay.infect(omicron)
'Wai Kay contracted Omicron'

>>> waikay.meet(ashish)

>>> ashish.infect(omicron)
'Ashish is immune to Omicron'
>>> waikay.infect(gamma)
'Wai Kay contracted Gamma'

>>> nitya.infect(alpha)
'Nitya contracted Alpha'
>>> nitya.meet(ashish)
>>> nitya.infect(omicron)
'Nitya is immune to Omicron'
>>> ashish.infect(alpha)
'Ashish is immune to Alpha'

>>> nitya.infect(delta)
'Nitya contracted Delta'
>>> nitya.infect(gamma)
'Nitya contracted Gamma'

>>> ashish.recover()
'Ashish recovers from COVID'
>>> ashish.recover()
'Ashish is COVID negative'
```

```
>>> ashish.infect(gamma)
'Ashish contracted Gamma'
```

— E N D O F P A P E R —