CS1010S — Programming Methodology
National University of Singapore

Practical Examination

20 November 2015

Time allowed: 2 hours

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. The time allowed for solving this quiz is **2 hours**.
3. The maximum score of this quiz is **30 marks**. Note that the number of marks awarded for each question **IS NOT** correlated with the difficulty of the question.
4. The total score for this quiz is capped at **20 marks** for those attempting the exam for the second time.
5. 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.
6. While you are also provided with the template `practical-template.py` to work with, your answers should be submitted on Coursemology.org. Note that you can run the test cases on Coursemology.org 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.
7. 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 rename your file `practical-exam-<mat no>.py` where `<mat no>` is your matriculation number place the file on the desktop. The invigilator will collect your files into the instructor PC remotely over the network.
8. Please note that it shall be your responsibility to ensure that your solution is submitted correctly to Coursemology (or appropriately named and placed on the desktop) at the end of the examination. Failure to do so will render you liable to receiving a grade of **ZERO** for the Re-Practical Exam, or the parts that are not uploaded correctly.
9. 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.

GOOD LUCK!

Question 1 : Seeing Double [10 marks]

A. Write a function `char_at` that take in as input, a character and a sentence, and outputs a list containing the positions (starting from 0) of the character in the sentence. The function should be case insensitive, i.e., uppercase and lowercase characters should be counted as the same.

Hint: the string functions `lower()` and `upper()` can be used to convert the case of a string. [4 marks]

Examples:

```
>>> sentence = 'The quick brown fox jumps over the lazy dog.'
>>> char_at('e', sentence)
[2, 28, 33]

>>> char_at('t', sentence)
[0, 31]

>>> char_at('7', sentence)
[]
```

B. Write a function `contains_duplicate` that takes a string and returns True if the string contains any duplicate characters, and False otherwise. The function should be case insensitive, i.e., uppercase and lowercase characters should be counted as the same.

You may assume that the `char_at` from part A is provided and correct. [4 marks]

Examples:

```
>>> contains_duplicate("abcdefg")
False

>>> contains_duplicate("ambidextrously")
False

>>> contains_duplicate("Double-dating")
True

>>> contains_duplicate("cs1010s")
True
```

C. Write a function `duplicate_within` that takes in a string and an integer i, and returns True if the there exists any duplicated characters in the string that are at most i characters apart, and False otherwise.The function should be case insensitive, i.e., uppercase and lowercase characters should be counted as the same.

You may assume that the `char_at` from part A is provided and correct. [2 marks]

Examples:

```
>>> duplicate_within("Double-dating", 6)
False  # The two 'd's are 7 characters apart

>>> duplicate_within("Double-dating", 7)
True

>>> duplicate_within("ballooning", 1)
True

>>> duplicate_within("ballooning", 0)
False

>>> duplicate_within("ambidextrously", 1)
False
```

Question 2 : Social Experiment [10 marks]

Important note: You are provided with a data file `connections.csv` for this question for testing, but your code should work correctly for *any* data file with the same format and it *will* be tested with other data files.

In a social experiment, participants are each given a mobile device which listens over Bluetooth for other such devices within range. When two participants are talking, their mobile devices will be within range of each other and maintain a connection between them. Researchers are interested in knowing the pattern of interactions among the participants.

You are provided with a data file containing the logs of the connections established between two mobile devices during the experiment. Each mobile device is given a unique ID, and each row represents a connection, with the starting time and ending time of the connection in 24-hour `HH:MM:SS` format. A pair of devices might make several connections with each other during the experiments, resulting in several rows in the data file.

A. Implement the function `get_connections` which takes in two integers, the ID of a mobile device and some time t in seconds and the name of the CSV file, and returns the number of connections established by the particular mobile node which lasted more than t seconds. You may assume that the mobile ID given is valid and found in the data provided. Also, the data provided are all within the same day. So there is no need to take care of the time passing midnight.

Some connections have the same start and end time. These are valid as the connection might only last a few milliseconds. These connections should be included in your computation and you can the connection time as 0 seconds.

Note also that each row represents a connection for both mobile devices, i.e., it should add to the connection count of both devices.

Hint: Suppose a string `s = "21:31:45"`, the function `s.split(":")` will return a list `["21", "31", "45"]`. You may wish to write a helper function that takes two strings in 24-hr time format and return the difference in seconds. [5 marks]

Sample execution:

```
>>> get_connections(34, 120, "connections.csv")
8

>>> get_connections(25, 60, "connections.csv")
27
```

B. Implement the function `get_summary` which takes as inputs an integer k and the name of the CSV file, and outputs a list of the top k mobile devices based on the number of connections made with each entry as a tuple containing:

`(device_ID, total_connections, average_time_per_connection)`

Top k meaning you should rank the devices according to their total connections and take the first k devices. If the $k+1$th device has the same total connections as the kth device, you should also include all devices with the same total connections as the kth device.

Some connections have the same start and end time. These are valid as the connection might only last a few milliseconds. These connections should be included in your computation and you should take the connection time as 0 seconds. [5 marks]

Sample execution:

```
>>> get_summary(1,"connections.csv")
[(34, 2759, 5.7140268213120695)]

>>> get_summary(3,"connections.csv")
[(34, 2759, 5.7140268213120695),
 (45, 2679, 4.889137737961926),
 (9, 2527, 6.694895132568263)]

>>> get_summary(7,"connections.csv")
[(34, 2759, 5.7140268213120695),
 (45, 2679, 4.889137737961926),
 (9, 2527, 6.694895132568263),
 (29, 2432, 4.198601973684211),
 (50, 2429, 9.234252778921366),
 (48, 2396, 4.770033388981636),
 (2, 2364, 7.076565143824027),
 (39, 2364, 4.876903553299492)]
# we get 8 entries because the 7th and 8th row are tied
# in number of connections
```

Question 3 : Inside Out [10 marks]

– BACKGROUND STORY (Okay to skip) –

A girl named Riley is born in Minnesota, and within her mind, five personifications of her core emotions—Joy, Sadness, Disgust, Fear, and Anger—come to life. The emotions live in Headquarters, Riley's conscious mind, where they influence Riley's actions and memories via a control console. Her new memories are housed in colored orbs, which are sent into storage at the end of every waking period. The most important or "core" memories are housed in a hub in Headquarters and power five "islands", each of which reflects a different aspect of Riley's personality. Joy acts as the dominant emotion to keep Riley in a happy state, but she and the others do not understand Sadness's purpose, other than making Riley cry.

When Riley is 11 years old, her family moves to San Francisco after her father gets a new job. However, their new home is lifeless, and their belongings are still in a moving truck that ends up getting lost somewhere in Texas. Joy becomes concerned when Sadness begins touching happy memories, causing them to turn sad, so she tries to keep Sadness isolated. However, on Riley's first day at her new school, Sadness accidentally causes Riley to cry in front of her class, creating a sad core memory. Joy attempts to dispose of the new core memory before it reaches the central hub, but she accidentally knocks the other core memories loose in her struggle with Sadness, shutting down the personality islands and making them unstable. Before Joy can put them back, she, Sadness, and the core memories are sucked out of Headquarters through the memory tube leading to the rest of Riley's mind. They end up in the labyrinthine storage area of Riley's long-term memories and set out to return to Headquarters.

Anger, Disgust, and Fear attempt to maintain Riley's emotional state in Joy's absence, but inadvertently distance Riley from her parents, friends, and hobbies, resulting in her personality islands slowly crumbling and falling into the Memory Dump, an abyss where faded memories are disposed and forgotten. Anger inserts an idea to run away to Minnesota into the control console, believing they can produce new happy core memories there,

but this only pushes Riley into a depression. Meanwhile, Joy and Sadness find Bing Bong, Riley's childhood imaginary friend, who is desperate to reconnect with her. He tells them they can get to Headquarters by riding the train of thought. After exploring different areas of Riley's mind, the three eventually catch the train, but it derails when another personality island falls.

As Riley prepares to board a bus bound for Minnesota, Joy attempts to use a "recall tube" to return to Headquarters, abandoning Sadness out of fear that her touch would change the happy core memories into sad ones if they travel in the tube together; however, the last personality island falls and breaks the tube, sending Joy into the Memory Dump along with Bing Bong. While despairingly looking through old memories, Joy discovers a sad memory in Riley's life that becomes happy when her parents and friends come to comfort her over losing a hockey game, causing her to realize Sadness's true importance: alerting others when Riley needs help or comforting.

Joy and Bing Bong try to use Bing Bong's discarded wagon rocket to get out of the Memory Dump, but after several failed attempts, Bing Bong realizes their combined weight is too much and jumps out and fades away, allowing Joy to escape. Joy uses various tools from Imagination Land to propel herself and Sadness to Headquarters, where they find that Anger's idea has disabled the control console, rendering Riley numb and apathetic. At Joy's encouragement, Sadness takes control and successfully removes the idea, reactivating the console and prompting Riley to return home.

As Sadness reinstalls the core memories, Riley arrives home and breaks down in tears, confessing to her parents that she misses her old life and she cannot keep pretending to be happy all the time. As her parents comfort and reassure her, Joy and Sadness work together to create a new amalgamated core memory, which creates a new personality island. A year later, Riley has adapted to her new home, and all her emotions now work together to help her lead a content, more emotionally complex life, with an expanded control console and additional personality islands produced by new amalgamated core memories comprising multiple emotions.

Source: Wikipedia

— START OF ACTUAL QUESTION —

Warning: Please read the entire question carefully and plan well before starting to write your code.

In this question, you will model and implement three classes: `Emotion`, `Memory` and `Island`. `Emotion` models the characters in Inside-Out. Using the console, Emotions can create memories which take after the emotion of the character at the console. `Memory` models the created memories and have a short title describing the memory. Memories can then be added to Islands (model by `Island`) to form the main personality.

An `Emotion` is created with one argument, its name which is a String. It supports the following **two** methods:

- `create()` which takes as input a string, and returns a new `Memory`. The `Memory` has a title which is the input string and an emotion which is the name of the `Emotion` that created it.
- `touch()` which takes as input a `Memory` and changes the emotion of the memory

to the name of the `Emotion`. If the emotion of the memory is already the same as the `Emotion`, then the string `'Nothing happens'` is returned. Otherwise, the string `'<memory.title> becomes <emotion>'` is returned.

A `Memory` is created with two arguments: a title (which is a string), and the emotion. `Memory` supports the following two methods:

- `get_title()` will return the title of the memory.
- `get_emotion()` will return the emotion of the memory.
- `forget()` will cause the memory to be forgotten. Once a memory is forgotten, it cannot be contained in an Island anymore. If the memory has already been forgotten, the string `'<memory.title> is already forgotten'` is returned. Otherwise, rhe string `'<memory.title> is forgotten'` is returned.

An `Island` is created with one argument, its name which is a string. It supports the following three methods:

- `add_memory()` which takes as input a memory, and does the following actions:
  - If the memory is already forgotten, then it cannot be added to itself, and the string `'<memory.title> has already been forgotten'` is returned.
  - If the memory is already part of an Island, i, then the string `'<memory.title> is already part of <i.name>'` is returned.
  - Otherwise, the memory will be added to the Island and the string `'<memory.title> added to <island.name>'` will be returned.
- `get_emotion()` takes no input, and returns the majority emotion of the memories that the Island contains. In other words, the emotion that belongs to the most number of memories it contains. If there are more than one majority, any of the emotion can be returned. If there are no memories in the Island, `None` is returned.
- `num_memories` takes no input and returns the number of memories that the Island contains.

Sample execution:

```
>>> joy = Emotion("happy")
>>> sadness = Emotion("sad")
>>> fear = Emotion("fearful")

>>> m1 = joy.create("First day at school")
>>> m1.get_emotion()
'happy'
>>> m1.get_title()
'First day at school'

>>> joy.touch(m1)
'Nothing happens'

>>> sadness.touch(m1)
'First day at school becomes sad'
```

```
>>> m1.get_emotion(),
'sad'

>>> m2 = joy.create("Scoring a goal")
>>> m3 = sadness.create("Failing an exam")

>>> friendship = Island("Friendship")
>>> friendship.add_memory(m1)
'First day at school added to Friendship'
>>> friendship.add_memory(m2)
'Scoring a goal added to Friendship'
>>> friendship.add_memory(m3)
'Failing an exam added to Friendship'
>>> friendship.num_memories()
3

>>> friendship.get_emotion()
'sad'   # Because there are 2 sad and 1 happy memories

>>> fear.touch(m1)
'First day at school becomes fearful'
>>> fear.touch(m3)
'Failing an exam becomes fearful'
>>> friendship.get_emotion()
'fearful'   # Because now there are 2 fearful and 1 happy memories

>>> m3.forget()
'Failing an exam is forgotten'
>>> friendship.get_emotion(),
'happy' # or 'fearful' since there are 1 of each left

>>> friendship.num_memories()
2

>>> m4 = joy.create("Playing ice-hockey")
>>> hockey = Island("Hockey")
>>> friendship.add_memory(m4)
'Playing ice-hockey added to Friendship'

>>> hockey.add_memory(m4)
'Playing ice-hockey is already part of Friendship'

>>> hockey.add_memory(m3)
'Failing an exam has already been forgotten'

>>> hockey.get_emotion()
None
```

— E N D O F P A P E R —