CS1010FC — Programming Methodology
National University of Singapore

Practical Examination

7 June 2014

Time allowed: 2 hours

Instructions (please read carefully):

1. This is **an open-book exam**.
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** and you will need to answer all three questions to achieve the maximum score. 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. Your answers should be submitted on Coursemology.org. You will be allowed to run some public tests cases to verify that your submission is correct. 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 on Coursemology.org. Do however ensure that you submit your answers correctly by running the test cases at least once.
6. In case there are problems with Coursemology.org, you are also provided with the template `practical-template.py` to work with. If Coursemology.org fails, you will be required to rename your file `practical-exam-<mat no>.py` where `<mat no>` is your matriculation number and upload the file to the Practical Exam folder in IVLE Workbin. You should only upload one file. If you upload multiple files, we will choose one at random for grading.
7. Please note that it shall be your responsibility to ensure that your solution is submitted correctly to Coursemology (or uploaded to IVLE, if there are problems) 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. You have been warned.
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.

GOOD LUCK!

Question 1 : Letter Rotation Mania [10 marks]

A. [**Warm Up: Letter Swap**] Write a function `swap` that if given a string will swap every alternate pair of letters. [5 marks]

Sample execution:

```
>>> swap("a")
a

>>> swap("ab")
ba

>>> swap("abc")
bac

>>> swap("abcc")
bacc

>>> swap("baabaablack")
abbaaalbcak
```

B. [**Full Scale Rotation**] Write the function `rotate` that takes in a string and two integers, *n* and *k*, respectively. `rotate` will operate on the string in chunks of *n* characters and do a *k*-character left shift for each chunk. In other words, `rotate(chars,2,1)` would be equivalent to `swap(chars)`. [5 marks]

Sample execution:

```
>>> rotate("baabaablack",2,1)
abbaaalbcak

>>> rotate("baabaablack",3,0)
baabaablack

>>> rotate("baabaablack",3,1)
aabaablabck

>>> rotate("baabaablack",3,2)
abaabaablck

>>> rotate("baabaablack",3,3)
baabaablack

>>> rotate("baabaablack",3,4)
aabaablabck
```

Question 2 : Financial Engineering 101 [10 marks]

You are given three CSV files: `sgd-rmb.csv`, `sgd-myr.csv`, `sgd-myr2.csv` which contain the exchange rates of Singapore Dollars (SGD) to Chinese Renminbi (RMB) and Malaysian Ringgit (MYR) in chronological order. Open the files to see their format. You are also provided with a helper function `read_csv` which you can find in the template file `practical-template.py`. We leave you to figure out what `read_csv` does. You are not required to use the helper function if you don't need it. You can assume that the data in the file is sorted in chronological order, i.e. that the earlier dates will always appear in an earlier line. Exchange rates for some dates might however be missing in the provided data files.

A. Write a function `average_rate(start_date, end_date, filename)` that finds the average exchange rate given by the currency data file `filename` between the dates `start_date` and `end_date` inclusive. You may assume that the exchange rates for the given dates `start_date` and `end_date` exist in the given file `filename`, and that `start_date` will come before `end_date`. [4 marks]

Sample execution:

```
>>> average_rate('11/26/2013', '1/31/2014', 'sgd-rmb.csv')
4.798673749999999

>>> average_rate('11/26/2013', '5/23/2014', 'sgd-myr.csv')
2.59280477124183

>>> average_rate('11/26/2013', '1/31/2014', 'sgd-myr.csv')
2.5881937931034478

>>> average_rate('11/26/2013', '1/31/2014', 'sgd-myr2.csv')
2.5880032142857137
```

B. Given a file `AB_filename` that contains exchange rates from currency *A* to currency *B* and another file `AC_filename` that contains the rates from currency *A* to currency *C*, write a function `convert(AB_filename, AC_filename)` that takes the two files and returns a dictionary whose keys are dates and whose values are the corresponding exchange rate from currency *B* to *C* for those dates. If either (or both) of the files does not contain the exchange rate for a particular date, the dictionary should not contain that date as a key. [3 marks]

Sample execution:

```
>>> myr_rmb = convert('sgd-myr.csv', 'sgd-rmb.csv')
>>> myr_rmb['4/29/2014']
1.9136195527357778

>>> '5/9/2014' in myr_rmb
False

>>> myr_rmb1 = convert('sgd-myr2.csv', 'sgd-rmb.csv')
```

```
>>> myr_rmb1['4/30/2014']
1.9237253527095648

>>> '5/9/2014' in myr_rmb1
True
```

C. Suppose that you are given some amount of currency *A* and can make up to 2 exchanges between currencies *A* and *B*. Write a function `trade(filename, amt)` that takes a file `filename` containing the exchange rates from currency *A* to *B* and an amount of currency *A* `amt` and returns the maximum amount of currency *A* obtainable by exchanging between the currencies. [3 marks]

Sample execution:

```
>>> trade('sgd-myr.csv',100)
102.88575760417642

>>> trade('sgd-myr2.csv',100)
102.88575760417642

>>> trade('sgd-rmb.csv',50)
51.53180938046961
```

Question 3 : X-Men: Telepaths [10 marks]

— BACKGROUND (Okay to skip) —

X-Men

The X-Men are a team of mutant superheroes in the Marvel Universe. They were created by writer Stan Lee and artist Jack Kirby, and first appeared in The X-Men #1 (September 1963). The basic concept of the X-Men is that under a cloud of increasing anti-mutant sentiment, Professor Xavier created a haven at his Westchester mansion to train young mutants to use their powers for the benefit of humanity, and to prove mutants can be heroes. Xavier recruited Cyclops, Iceman, Angel, Beast, and Marvel Girl, calling them "X-Men" because they possess special powers due to their possession of the "X-gene," a gene which normal humans lack and which gives mutants their abilities. Early on, however, the "X" in X-Men stood for "extra" power which normal humans lacked. It was also alluded to that mutations occurred as a result of radiation exposure. The comic was partially inspired by the African-American civil rights movement of the 1960s.

The first issue also introduced the team's archenemy, Magneto, who would continue to battle the X-Men for decades throughout the comic's history, both on his own and with his Brotherhood of Mutants (introduced in issue #4). The X-Men universe also includes such notable heroes as Wolverine, Storm, Colossus, Emma Frost, Gambit, Nightcrawler, Rogue, Psylocke and Shadowcat. Besides the Brotherhood of Mutants, other villains that the X-Men have fought include the Sentinels, Apocalypse, Mister Sinister, the Marauders, the Acolytes, the Hellfire Club, the U-Men, and the Purifiers.

Telepathy

Telepathy (from the Ancient Greek , tele meaning "distant" and παθος, pathos or -patheia meaning "feeling, perception, passion, affliction, experience") is the purported transmission of information from one person to another without using any of our known sensory channels or physical interaction. The term was coined in 1882 by the classical scholar Frederic W. H. Myers, a founder of the Society for Psychical Research, and has remained more popular than the earlier expression thought transference.

Some fictional telepaths possess mind control abilities, which can include "pushing" thoughts, feelings, or hallucinatory visions into the mind of another person, causing pain, paralysis, or unconsciousness, altering or erasing memories, or completely taking over another person's mind and body (similar to spiritual possession). Examples of this type of telepath include Professor Xavier, Psylocke, Jean Grey, Emma Frost, and numerous other characters in the Marvel Universe, along with Matt Parkman from the television series Heroes.

– Source: Wikipedia

In this question, you will model telepathic interactions by implementing the class `Mutant` and its subclass `Telepath`. For simplicity, we will assume that all telepaths have full mind control abilities in the sense of being able to control the actions of another mutant. In particular, a mind-controlled telepath can be made to mind-control another mutant. A *mutant* may be a telepath or not.

Telepaths can directly mind-control only one mutant at a time. When this happens, we say that that mutant has become the telepath's *slave* and the telepath has become the mutant's *master*. A telepath T may use a slave telepath to mind-control another telepath and then use that telepath to capture yet another telepath and so on, creating a chain of slaves with T as their *mastermind*. No cyclic mind-control is allowed, i.e. a slave telepath cannot be made to control a telepath that already has the same mastermind as himself.

A mastermind may also choose to release a slave from mind-control. If the chosen slave is a telepath, then the slave's own slaves are all set free from any mind-control.

Telepaths have varying telepathic strengths. Non-telepathic mutants are defenseless against a telepath but a telepath can only mind-control a (strictly) weaker telepath. The exception is when the target mutant is being mind-controlled, in which case a telepath has to be stronger than the mutant's mastermind in order to wrest control of the mutant from him. When a telepath changes masters or becomes mind-controlled, the telepath retains control over his slaves, if any.

A `Mutant` is initialised with the mutant's name and has at least the following **three** methods:

1. `get_name()` returns the name of the mutant.
2. `get_master()` returns the mutant himself if he is not being mind-controlled, otherwise it returns the `Telepath` controlling it.
3. `get_mastermind()` returns the mutant herself if she is not being mind-controlled, otherwise it returns the mastermind `Telepath` controlling it.

A `Telepath` is a special type of `Mutant`. Its initialiser takes as arguments a name and a number representing telepathic strength, and has at least the following **three** methods:

1. `get_slave()` returns the `Mutant` or `Telepath` object the telepath is controlling. If the telepath is controlling no one, return `None`.
2. `mind_control(m)` causes the telepath to mind-control the `Mutant` (or `Telepath`) `m`. If the telepath does not have a slave, then the telepath mind-controls `m` directly. Otherwise, the telepath uses the last slave in his chain of slaves to do the mind-controlling, if that last slave is a telepath. If the last slave is a non-telepath mutatnt, nothing happens. The success of the mind-control attempt is contingent on the rules above.
3. `release(*m)` causes the telepath to release the `Mutant` (or `Telepath`) `m` from mind-control. If `release` is called with zero arguments, the telepath releases all its slaves. No action is taken if `m` is not one of the telepath's slaves to begin with. Releasing a mind-controlled telepath who controls some other telepath or mutant will also free the controlled mutant as well.

If the `Telepath` is enslaved, calling `mind_control` or `release` should do nothing.

Implement the `Mutant` and `Telepath` classes. A sample execution can be found on the next page.

Sample execution (newlines are added for readability):

```
>>> logan = Mutant("Wolverine")
>>> charles = Telepath("Prof. X", 10)
>>> jean = Telepath("Phoenix", 9)
>>> psylocke = Telepath("Psylocke", 8)
>>> emma = Telepath("Frost", 7)

>>> psylocke.mind_control(logan) # [P -> L]
>>> logan.get_master().get_name()
Psylocke
>>> psylocke.get_slave().get_name()
Wolverine

>>>  # === jean steals logan from psylocke ===
>>> jean.mind_control(logan) # [J -> L]
>>> logan.get_master().get_name()
Phoenix
>>> jean.get_slave().get_name()
Wolverine
>>> psylocke.get_slave()
None

>>> charles.mind_control(jean) # [C -> J -> L]
>>> logan.get_master().get_name()
Phoenix
>>> jean.get_master().get_name()
Prof. X
>>> charles.get_slave().get_name()
Phoenix

>>> # === psylocke fails to steal jean ===
>>> psylocke.mind_control(jean) # [C -> J -> L]
>>> jean.get_master().get_name()
Prof. X

>>> # === psylocke fails to control charles ===
>>> psylocke.mind_control(charles) # [C -> J -> L]
>>> charles.get_master().get_name()
Prof. X

>>> charles.release(logan) # [C -> J]
>>> charles.get_slave().get_name()
Phoenix
>>> jean.get_slave().get_name()
None
```

```
>>> logan.get_master().get_name()
Wolverine

>>> psylocke.mind_control(emma) # [P -> E, C -> J]
>>> emma.get_master().get_name()
Psylocke

>>> charles.mind_control(psylocke) # [C -> J -> P -> E]
>>> emma.get_mastermind().get_name()
Prof. X
>>> psylocke.get_mastermind().get_name()
Prof. X
>>> jean.get_mastermind().get_name()
Prof. X
>>> charles.get_mastermind().get_name()
Prof. X

>>> # === jean cannot release anybody ===
>>> jean.release(psylocke) # [C -> J -> P -> E]
>>> psylocke.get_master().get_name()
Phoenix

>>> # === emma cannot act ===
>>> emma.mind_control(logan) # [C -> J -> P -> E]
>>> logan.get_master().get_name()
Wolverine

>>> # === failed cyclic control attempt ===
>>> charles.mind_control(jean) # [C -> J -> P -> E]
>>> emma.get_slave()
None

>>> # === release all ===
>>> charles.release(jean)
>>> charles.get_master().get_name()
Prof. X
>>> jean.get_master().get_name()
Phoenix
>>> emma.get_master().get_name()
Frost
>>> psylocke.get_master().get_name()
Psylocke
>>> logan.get_master().get_name()
Wolverine
```