CS1010X — Programming Methodology
National University of Singapore

Practical Examination

6 June 2020

Time allowed: 2 hours

Instructions (please read carefully):

1. This is an **open-book** and an **open Internet** exam. However, **no credit will be given if you cut-and-paste code from the web**. We will be running plagiarism software to compare your submissions to your classmates' submissions and whatever relevant code we find available on websites like GitHub, Leetcode, Stack Overflow, Tutorials Point, etc. Any suspect plagiarism cases will be subject to further investigation with serious consequences if found guilty.
2. You are to do your work without any assistance from another intelligent human being – if found otherwise, you will receive **ZERO** for the practical exam and will be subject to other disciplinary actions.
3. This practical exam consists of **three** questions. The time allowed for solving this quiz is **2 hours**.
4. 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.
5. Your answers should be submitted on Coursemology.org **BEFORE** the end of the exam. If you have any submissions timestamped with a time after the exam has ended, your submission for that question will not be graded. Remember to **finalize** your submissions before the end of the exam.
6. 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 up to a **maximum of 5 times** 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.
7. Please note that it shall be your responsibility to ensure that your solution is submitted correctly to Coursemology 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 : ET Arithmetic [10 marks]

Any positional number system can be characterized by 2 parameters, a `base`, which is the total number of unique digits it uses to represent numbers and the `digits` themselves. For example, the decimal system has a `base=10` because it uses 10 digits `('0', '1', '2', '3', '4', '5', '6', '7', '8', '9')` to represent all numbers. The binary system (`base=2`) uses only 2 digits `('0', '1')` to represent all numbers.

In this question, we will characterize each ET (Extra-terrestrial) number system with a unique `mapping` which will basically be an ordered tuple of the digits used by that number system. For example, the `mapping` of the decimal system is `('0', '1', '2', '3', '4', '5', '6', '7', '8', '9')` and the mapping of the binary system is `('0', '1')`. Naturally, the `base` of such number systems will be equal to `len(mapping)`.

[HINT]
This question extends the logic introduced in Extra Practice on Higher Order Functions : Q5 Binary to any base decimal [Here are the links: `question on coursemology` and the `Forum post`].

A. Your first task is to write a function `ET_numbers` that takes in two parameters: `number` and `mapping`. The function must return the 'ET number' representation of `number` (which is a decimal integer) based on the `mapping` passed to it. You can assume that the mapping will always have at least two symbols in it (at least base 2). The sample execution is shown below.

[5 marks]

Sample execution:

```
>>> ET_number(5, ('0','1','2','3','4','5','6','7','8','9'))
'5'

>>> ET_number(20, ('9','8','7','6','5','4','3','2','1','0'))
'79'

>>> ET_number(10, ('0','1','2','3','4','5'))
'14'

>>> ET_number(6, ('0','4'))
'440'

>>> ET_number(5, ('1', '0'))
'010'

>>> ET_number(10, ('a', 'b', 'c'))
'bab'
```

B. Your second task is to write the function `max_ET_number` that takes a tuple of `ET_numbers` and their corresponding `mapping` and returns the ET number with the highest value. You can assume that each ET_number in `ET_numbers` is of some non-negative value when converted into a decimal integer. Sample execution is shown below. [5 marks]

Sample execution:

```
>>> max_ET_number((‘1’,‘2’,‘3’,‘4’,‘5’), (‘0’,‘1’,‘2’,‘3’,‘4’,‘5’,‘6’,‘7’,‘8
‘5’

>>> max_ET_number((‘12’,‘34’,‘42’,‘58’), (‘0’,‘1’,‘8’,‘3’,‘5’,‘4’,‘6’,‘7’,‘2
‘42’

>>> max_ET_number((‘19’,‘20’,‘21’), (‘0’,‘2’,‘1’,‘3’,‘4’,‘5’,‘6’,‘7’,‘8’,‘9’
‘19’

>>> max_ET_number((‘14’,‘15’),(‘0’,‘1’,‘2’,‘3’,‘5’,‘4’))
‘14’

>>> max_ET_number((‘707’,‘700’,‘770’),(‘0’,‘7’))
‘770’

>>> max_ET_number((‘0’, ‘4’, ‘40’, ‘44’, ‘4004’, ‘4040’),(‘0’,‘4’))
‘4040’

>>> max_ET_number((‘317’,‘311’,‘713’,‘413’),(‘7’,‘1’,‘3’,‘4’))
‘413’

>>> max_ET_number((‘aba’, ‘abc’, ‘ca’, ‘cb’), (‘a’, ‘b’, ‘c’))
‘cb’
```

Question 2 : Tesla Stocks [10 marks]

On 1st May, 2020, Elon Musk famously tweeted "The Tesla stock is too high imo", following which the Tesla stock tanked. His tweet reduced the price of a single Tesla share by 10% costing his company billions. This is not the first time this has happened. There have been multiple ocassions on which his tweets seemed to have clearly impacted the stock prices of Tesla. Elon has even been sued for tweeting about his company and inadvertently effecting its stock prices before.

In this question, we will be investigating this a little further. You have been given two comma-separated files (`.csv`) named `tweets.csv` and `TSLA.csv` which contain tweets scraped off of Elon Musk's twitter and Tesla's share prices since it went public in 2010, respectively. Both these files have already been sorted by date for you. Each line in `tweets.csv` has the following data:

1. `handle` : The twitter handle of the tweeter
2. `name` : Name of the tweeter
3. `content` : Contents of the tweet (text only)
4. `replies` : Replies to the tweet
5. `retweets` : Number of re-tweets
6. `favorite` : Number of likes
7. `date` : Date of the tweet (mm/dd/yyyy)

And each line in the file `TSLA.csv` has the following information:

1. `date` (mm/dd/yyyy)
2. `stockprice`

The function to help you read these `.csv` files (`readcsv()`) has been provided to you in the template file.

You can also utilize the `datetime` library in python to convert a date string to a date object, so that you can perform comparisons (`==, <, >, <=, >=`) and arithmetic on them. For example:

```
import datetime
# converting a string to a datetime object:
date = datetime.datetime.strptime( "10/31/1985", "%m/%d/%Y" )

# adding 5 days to a given date (which is a datetime object):
new_date = date + datetime.timedelta(days=5)
```

A. [WARM UP] Getting tweets by date

Write a function `get_tweet_by_date(date)` that takes in a `date` (string) and returns a tuple with the `content` of all the tweets made on that date. If there were no tweets on that day, your function should return an empty tuple.

[3 Marks]

Sample Execution:

```
>>> get_tweet_by_date('12/21/2019')
('Great show  https://t.co/12rguHHpgz',
'Holiday gift ideas https://t.co/uBBofvkYAI')

>>> get_tweet_by_date('5/1/2020')
('Now give people back their FREEDOM',
'I am selling almost all physical possessions. Will own no house.',
'Tesla stock price is too high imo',
'And the rocket's red glare, the bombs bursting in air',
'Rage, rage against the dying of the light of consciousness')

>>> print(get_tweet_by_date('12/12/2001')
()

>>> get_tweet_by_date('5/21/2021')
()
```

B. Market reaction to tweets

Write a function `tweeteffect(date)` that takes a `date` (string) and returns a tuple with all the tweets made that day and a list with the stock price of TESLA (floats) for the next 5 days. Note that the number of stock prices found may be less than 5 because stock market may not be opened on some dates. If there were no tweets made on the given `date`, your function should return `None`.

[3 Marks]

Sample Execution:

```
>>> tweet_effect('5/8/2013')
('Just want to say thanks to customers & investors that
took a chance on Tesla through the long, dark night.
We wouldn't be here without you.',
[55.790001, 69.400002, 76.760002, 87.800003])

>>> tweet_effect('3/23/2017')
None

>>> tweet_effect('7/14/2019')
('To Infinity and Beyond! https://t.co/dgysTBqWfV',
[253.5, 252.380005, 254.860001, 253.539993, 258.179993])
```

C. Money tweets

Write a function `money_tweets(start_date, end_date)` that takes a `start_date` (which is a string) and an `end_date` (also a string) to return a tuple with the (one or more) tweets (on a same day) that caused the highest market fluctuations in a five-day period following the day on which those tweets were tweeted. Market fluctuation in a 5 day period can simply be calculated as the difference between the highest and lowest price of the TESLA stock during that period. If there were no tweets made in the specified time span, your function must return `None`.

[4 Marks]

Sample Execution:

```
>>> costly_tweets('5/12/2020', '5/21/2020')
('Ice cream sundae in a martini glass
https://t.co/zAVFlOsYkM', 'Super exciting day coming up!
https://t.co/7ZdFsJE9zR', 'https://t.co/lQWpSwtRj7'),
22.669983000000002)

>>> costly_tweets('4/29/2020', '5/1/2020')
('FREE AMERICA NOW', 'Give people their freedom back!
https://t.co/iG8OYGaVZ0', 'Bravo Texas!
https://t.co/cVkDewRqGv'), 99.19000299999993)
```

Question 3 : TOY TRAIN [10 marks]

In this question, we will be modelling a simple train. We will represent our train on a grid (as shown in Figures 1 and 2), with each engine or carriage taking up one cell. Your task is to simulate the movement of this train on the grid given a set of movements fed to the engine.

To achieve this, you need to implement the two components that make up a train – a `carriage` class and an `engine` class (which is a child class of `carriage`).

The `carriage` object must have the following functions associated with it:

1. `carriage(x, y)` : The constructor of this class will take 2 coordinates, which will determine the starting position (x, y) of the carriage.
2. `get_x()` : returns the x coordinate of the carriage
3. `get_y()` : returns the y coordinate of the carriage
4. `get_pos()` : returns the carriages position (x, y) as a tuple.
5. `attach(carriage)` : Attaches another carriage to the given carriage. A carriage can only be attached to one other carriage and pulls whichever carriage is attached to it. Two carriages can be attached only if they are next to each other in the grid. In the event of a successful attachment, the function must return the string "`Attached!`", and the string "`Can't attach.`" otherwise.

The `engine` object will have one additional function associated with it:

1. `move(track)` : This function will take a single argument, `track`, which is a string of characters (directions) and move the engine according to these directions. The engine can move in the up, down, right and left direction on the grid:
   - `'u'`: movement in the positive y direction
   - `'d'`: movement in the negative y direction
   - `'r'`: movement in the positive x direction
   - `'l'`: movement in the negative x direction

   This move function should not only change the position of the engine, but also suitably effect the position of the carriages attached to it. This function should also check that the train is not colliding into itself, as shown in Figures 2. Given a string of movements, the move function should follow these movements as long as they are possible without collision. That is, if a collision happens, for example, on the third movement, the function should complete the first and second movements, and then return the string "`Collision!`". For simplicity, we will only be detecting collisions of the train with itself, and not other stray carriages/engines on the grid. If the function was able to carry out the given string of movements successfully, it should return `None`.

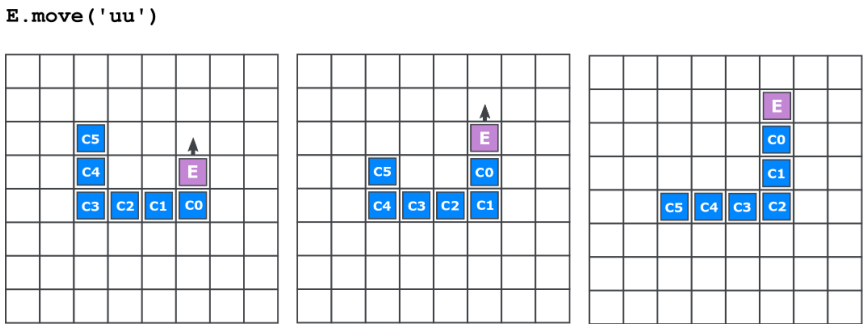


Figure 1: How the train is expected a move in the grid. Basically, the carriages should follow wherever the engine has been as shown from the left picture to the middle picture (after the first 'u' move) then to the right picture (after the second 'u' move).

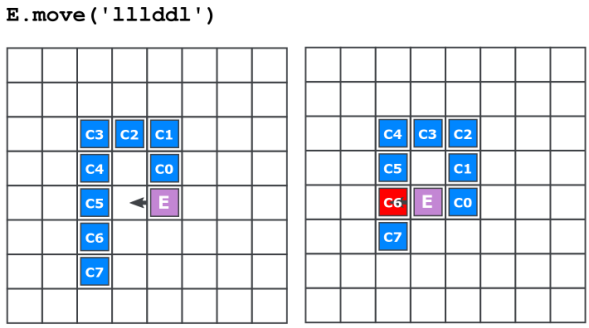
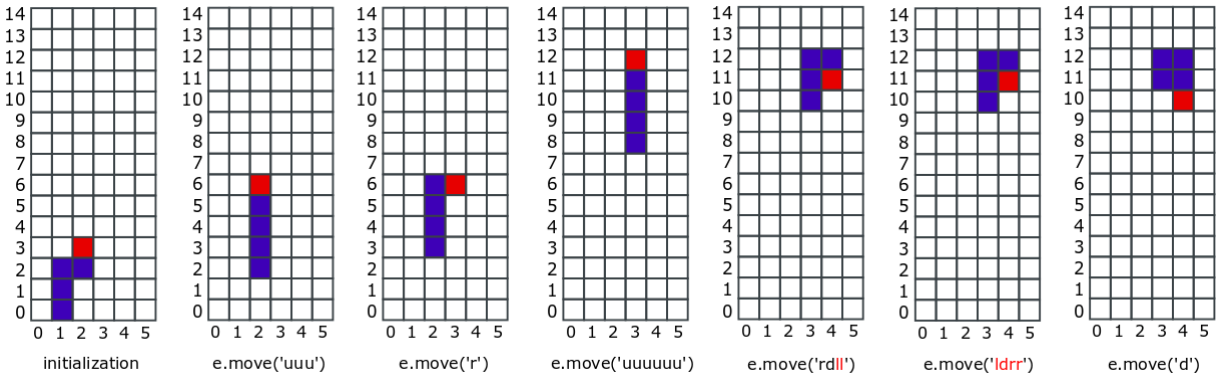


Figure 2: How a train can collide with its own carriages. In such a scenario, all movements leading up to the collision should be carried out.

Sample Execution:

```
>>> c0 = carriage(1,0)
>>> c1 = carriage(1,1)
>>> c2 = carriage(1,2)
>>> c3 = carriage(2,2)
>>> c4 = carriage(3,4)
>>> e  = engine(2,3)

>>> c1.get_x()
1
>>> c3.get_y()
2
>>> c0.get_pos()
(1,0)
```




```
# Attaching carriages together to build the train
>>> e.attach(c3)
"Attached."
>>> c3.attach(c2)
"Attached."
>>> c2.attach(c1)
"Attached."

# c1 and c4 are not adjacent
>>> c1.attach(c4)
"Can't attach."

>>> c1.attach(c0)
"Attached."

# getting the position of the engine and all the carriages
>>> (e.get_pos(), c3.get_pos(), c2.get_pos(), c1.get_pos(), c0.get_pos())
((2, 3), (2, 2), (1, 2), (1, 1), (1, 0)))

>>> e.move('uuu')
None
>>> (e.get_pos(), c3.get_pos(), c2.get_pos(), c1.get_pos(), c0.get_pos())
(2, 6), (2, 5), (2, 4), (2, 3), (2, 2))

>>> e.move('r')
None
>>> (e.get_pos(), c3.get_pos(), c2.get_pos(), c1.get_pos(), c0.get_pos()
((3, 6), (2, 6), (2, 5), (2, 4), (2, 3))

>>> e.move('uuuuuu')
None
>>> (e.get_pos(), c3.get_pos(), c2.get_pos(), c1.get_pos(), c0.get_pos())
((3, 12), (3, 11), (3, 10), (3, 9), (3, 8))
```

```
>>> e.move('rdll')
"Collision!"
>>> (e.get_pos(), c3.get_pos(), c2.get_pos(), c1.get_pos(), c0.get_pos())
((4, 11), (4, 12), (3, 12), (3, 11), (3, 10))

>>> e.move('ldrr')
"Collision!"
>>> (e.get_pos(), c3.get_pos(), c2.get_pos(), c1.get_pos(), c0.get_pos())
((4, 11), (4, 12), (3, 12), (3, 11), (3, 10))

>>> e.move('d')
None
>>> (e.get_pos(), c3.get_pos(), c2.get_pos(), c1.get_pos(), c0.get_pos())
((4, 10), (4, 11), (4, 12), (3, 12), (3, 11))
```

— E N D O F P A P E R —