NATIONAL UNIVERSITY OF SINGAPORE

CS1010S—Programming Methodology

2018/2019 Semester 2

Time Allowed: 2 hours

INSTRUCTIONS TO STUDENTS

1. Please write your Student Number only. Do not write your name.
2. The assessment paper contains **FIVE (5) questions** and comprises **TWELVE (12) pages** including this cover page.
3. Weightage of each question is given in square brackets. The maximum attainable score is 100.
4. This is a **CLOSED** book assessment, but you are allowed to bring **ONE** double-sided A4 sheet of notes for this assessment.
5. Write all your answers in the space provided in the **ANSWER BOOKLET**.
6. You are allowed to write with pencils, as long as it is legible.
7. **Marks may be deducted** for i) unrecognisable handwriting, and/or ii) excessively long code. A general guide would be not more than twice the length of our model answers.
8. Common `List` and `Dictionary` methods are listed in the Appendix for your reference.

This page is intentionally left blank.

It may be used as scratch paper.

Question 1: Python Expressions [30 marks]

There are several parts to this problem. Answer each part **independently and separately**. In each part, one or more Python expressions are entered into the interpreter (Python shell). Determine the response printed by the interpreter for the final expression entered and **write the exact output in the answer box**. If the interpreter produces an error message, or enters an infinite loop, explain why and **clearly state the responsible evaluation step**.

The code is replicated on the answer booklet. You may show your workings **outside the answer box** in the space beside the code. Partial marks will be awarded for workings if the final answer is wrong.

A.
```
s = 'CS1010S-S0-C001'
d = {}
for i in range(len(s)):
    d[s[i]] = s[i%5]
print(d)
```
[5 marks]

B.
```
d = {1:2, 2:4, 3:6, 4:8,
     5:0, 6:2, 7:4, 8:6,
     9:8, 0:3}
n = 1
for i in range(5):
    n = d[n]
print(n)
```
[5 marks]

C.
```
a = [1, 2]
a += [a]
b = a.copy()
a[2] = 0
print(a)
print(b)
```
[5 marks]

D.
```
def boo(x):
    try:
        i, j = int(x)
        print(i / j)
    except ZeroDivisionError:
        print('Why zero?')
    except ValueError:
        print('Wrong Value!')
    except IndexError:
        print('Wrong Index!')
    except TypeError:
        print('Wrong Type!')
boo(['ten', 0])
```
[5 marks]

E.
```
def g(x):
    if x:
        return (x[-1], g(x[1:-1]), x[0])
    else:
        return ()
print( g([1, 2, 3, 4, 5]) )
```
[5 marks]

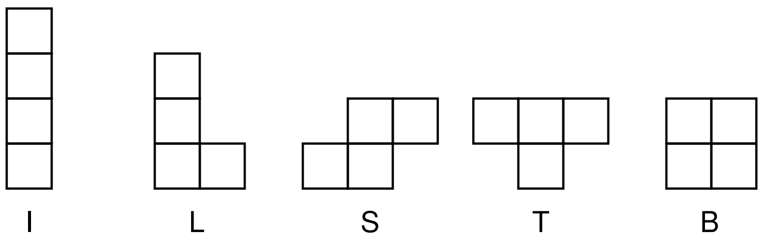
F.
```
def bar(f, g):
    return lambda x: (lambda y: f(x))(g(x))
print( bar(lambda x:x+1, lambda x:x*2)(5) )
```
[5 marks]

Question 2: Tetrominos [31 marks]

Advisory: This is a long question with several parts. You are advised to read the entire question before attempting.

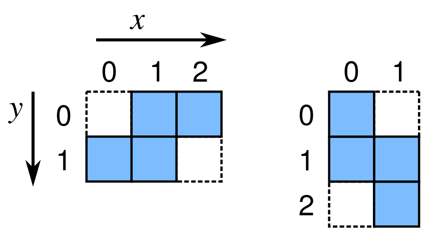
A tetromino is a geometric shape composed of four squares, connected orthogonally. A popular use of tetrominoes is in the video game Tetris, which refers to them as tetriminos.

Source: Wikipedia



The five possible configurations of tetrominos, assuming rotations and flips are congruent.

There are several ways to represent a tetromino in Python. In our representation, each tetromino is tightly bounded in a grid, with the top left corner being coordinates (0, 0). A tetromino is then represented in Python as a list of (x, y) coordinates that is occupied by each square of the tetromino. Note that the order of the list does not matter.



For example, the S tetromino shown above can be represented as a list of tuples `[(0,1), (1,0), (1,1), (2,0)]` or `[(0,0), (0,1), (1,1), (1,2)]` depending on its orientation.

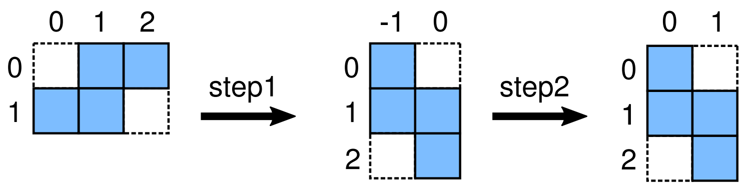
The function `rotate` takes as input a tetromino, and rotates it 90 degrees clockwise. Thus, calling rotate on an S tetromino repeatedly will toggle it between the two representations shown above.

A. Joseph Saelee thinks the coordinates in the list representing a tetromino should be created using lists instead of tuples as calling `rotate(tetromino)` should modify the input tetromino to a different configuration.

Do you agree with him? Explain why. [3 marks]

To rotate a tetromino, the first step is to rotate it geometrically about the origin (0, 0) This will result in the coordinates of some squares becoming negative. The second step is to normalize, or translate the coordinates to standard position with a tightly bounded grid at (0, 0).

The illustration below shows how an S tetromino is rotated clockwise.



B. Implement the function `step1` that takes in a tetromino and performs the first step of the rotation as described, i.e. the (x,y) coordinate of each square transforms to $(-y,x)$.

Note that the function should modify the input tetromino and not return anything.

You may represent the coordinates using lists or tuples. [5 marks]

C. Implement the function `step2` that takes in a tetromino and performs the second step of the rotation described above, i.e. each coordinate is translated such that the top left corner of the tightly bound grid is $(0,0)$.

As before, the function should modify the input tetromino and not return anything. Remember to maintain the same representation for coordinates as used in part B. [5 marks]

D. Suppose `s` is an S tetromino, and the following statement are executed:

```
>>> s = [(0,1), (1,0), (1,1), (2,0)]  # or [[0,1], [1,0], [1,1], [2,0]]
>>> t = s.copy()                       # using lists

>>> step1(s)          # as implemented
>>> step2(s)          # in 2B and 2C
```

Draw the box-pointer diagram of `s` and `t` according to your implementation in 2B and C. For clarity, you may place integers inside the box if you wish. [4 marks]

E. The function compose is defined as follows:

```
def compose(f, g):
    return lambda x: f(g(x))
```

Since the `rotate` is composed of the functions `step1` and `step2`, Joseph defined `rotate` as such: `rotate = compose(step1, step2)`.

However, he is not able the get `rotate` to work as described. Explain why this is so. [3 marks]

Suppose the rows of a playing grid of a game using tetrominos is modelled using lists, and the entire grid is represented as a list of rows, with the **first row (i.e. index 0) representing the top-most row**. Each element in a row is a `Bool`, with `True` indicating that it is occupied by a square, and `False` otherwise.

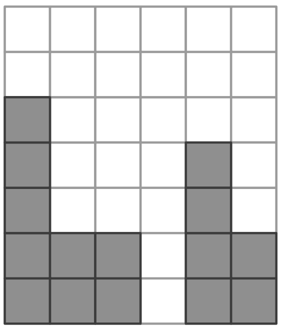


Fig A. The grid. Occupied blocks are shaded.

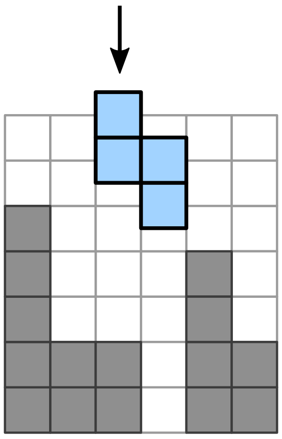


Fig B. S-tetromino is dropped at col 2.

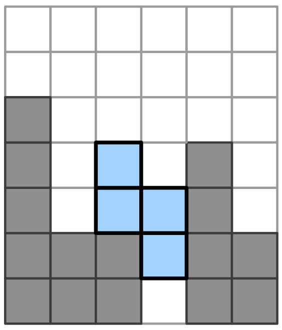


Fig C. S-tetromino is finally placed at row 3.

The grid in **Fig A** above is represented as:

```
[[False, False, False, False, False, False],
 [False, False, False, False, False, False],
 [True,  False, False, False, False, False],
 [True,  False, False, False, True,  False],
 [True,  False, False, False, True,  False],
 [True,  True,  True,  False, True,  True],
 [True,  True,  True,  False, True,  True]]
```

F. A tetromino can be placed into the grid as long as it does not overlap with an existing occupied square **and** does not exceed the height of the grid. For example, as shown **Fig C**, the S tetromino in vertical orientation can be placed into row 3, in column 2 (using the tetromino top-left origin as reference).

Implement a function `can_place` which takes in a tetromino, a grid, and the row and column to insert the given tetromino into the grid, The function returns `True` if the placement is possible, `False` otherwise. The input grid should not be modified and you may assume placement will not exceed the width of the grid. [5 marks]

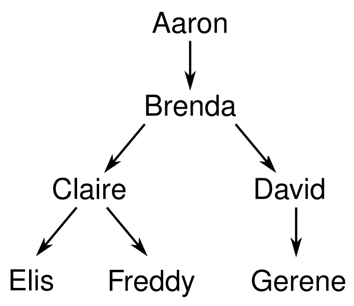
G. In a game, a tetromino is dropped from the top of the grid at a particular column, and falls downward until it comes to a rest on top of the existing squares on the grid.

For example, **Fig B** depicts the S tetromino being dropped in column 2 and comes to its final resting place at row 3 as shown in **Fig C**.

Implement a function `drop` which takes in a tetromino, a grid and a column. If there is sufficient headroom to place the new tetromino, the function returns `True` and **updates** the occupied squares in the grid to include this new tetromino. Otherwise, `False` is returned **without** any modifications to the grid. [6 marks]

Question 3: Family Business [19 marks]

A family tree is a tree that records the genealogy of a family. One such example is a descendancy chart, which depicts all the descendants of an individual at the top. For example:


$$\begin{tikzcd}
 & & \text{Aaron} \arrow[d] & \\
 & & \text{Brenda} \arrow[dl] \arrow[dr] & \\
 & \text{Claire} \arrow[dl] \arrow[d] & & \text{David} \arrow[d] \\
\text{Elis} & \text{Freddy} & & \text{Gerene}
\end{tikzcd}$$

Each line connects a parent to a child, thus the tree shows all the descendants of Aaron. We can record these parental relationships using a dictionary where the keys will be the child and the value be the respective parent. The tree shown above would be represented using the following dictionary:

```
parents = {'Brenda': 'Aaron', 'Claire': 'Brenda', 'David': 'Brenda',
           'Elis': 'Claire', 'Freddy': 'Claire', 'Gerene': 'David'}
```

A. Implement a function `is_descendent` that takes in a parental dictionary, along with two names, *a* and *d* and returns `True` if *d* is a descendent of *a*, and `False` otherwise.

For example: `is_descendent(parents, 'Aaron', 'Elis')` returns `True` and `is_descendent(parents, 'Claire', 'Gerene')` returns `False`. [5 marks]

B. Sometimes, we need to obtain the children of a particular person in the family tree. It would be helpful if we have an "inverse" dictionary that maps a person to a list of their direct children, i.e. a *parent→children* dictionary.

For example, the "inverse" (*parent→children*) dictionary of `parents` would be:

```
>>> convert(parents)
{'Aaron': ['Brenda'], 'Brenda': ['Claire', 'David'],
 'Claire': ['Elis', 'Freddy'], 'David': ['Gerene'],
 'Elis': [], 'Freddy': [], 'Gerene': []}
```

Implement the function `convert` which takes in a *child→parent* dictionary, and returns the "inverse" which is a *parent→children* dictionary as described. [5 marks]

C. Re-implement the function `is_descendent` to take in a *parent→children* dictionary mapping instead of a *child→parent* dictionary as its first input.

[6 marks]

D. Suppose in the given family tree, cousins Freddy and Gerene marries and have a child. How does that affect the *parent→children* and *child→parent* dictionaries? [3 marks]

Question 4: Apple [16 marks]

Consider the following OOP implementation:

```python
class StorageDevice:
    def __init__(self, memory, *args):
        super().__init__(*args)
        self.memory = memory
        self.used = 0
        self.items = []  # items stored in memory

    def available_storage(self):
        return self.memory - self.used - len(self.items)


class PhoneDevice:
    def __init__(self, number, *args):
        super().__init__(*args)
        self.number = number

    def call(self, number):
        print('Calling number', number)

    def receive(self):
        print('Receiving Call...')


class MusicPlayer(StorageDevice):
    def __init__(self, memory):
        super().__init__(memory)

    def play_song(self, song):
        if song in self.items:
            print("Playing " + song)
        else:
            print(song + " not found")


class IPod(MusicPlayer):
    def __init__(self, memory):
        super().__init__(memory)

    def backup_songs(self):
        print("Connect to iTunes to back up songs")
```

A. State the output when each of the following expressions is executed in the Python shell. If an error occurs, you may just state "ERROR" and explain why.

```
>>> ipod = IPod(32)
>>> ipod.available_storage
>>> ipod.play_song("Somewhere Out There")
>>> ipod.call(65169121)
```

[3 marks]

Currently our iPod has no songs loaded and we wish to load some songs into it. Consider the following execution and the output:

```
>>> ipod = IPod(32)
>>> ipod.store_song("Our Song", 14)
Our Song stored
17 memory left

>>> ipod.store_song("Never Say Never", 10)
Never Say Never stored
6 memory left

>>> ipod.play_song("Our song")
Playing Our song

>>> ipod.store_song("Never Enough", 6)
Not enough memory

>>> ipod.play_song("Never Enough")
Never Enough not found
```

B. In which class should the function `store_music` be placed? [2 marks]

C. Provide an implementation for the function `store_music` that will produce the same behaviour as the execution trace shown above. You may assume it will be added into the class you decided in 4B. [5 marks]

D. An iPhone can both play music and also make phone calls. It can also search the web using Google! Below is a sample execution:

```
>>> iphone = IPhone(256)
>>> iphone.call(98765432)
Calling number 98765432

>>> iphone.play_song('Sha La La')
Sha La La not found

>>> iphone.search("How to program with Python?")
Searching How to program with Python?

>>> iphone.backup_songs()
AttributeError: 'IPhone' object has no attribute 'backup_songs'
```

Provide an implementation of the class `IPhone`. You should use OOP practices whenever possible. [6 marks]

Question 5: 42 and the Meaning of Life [4 marks]

Either: (a) explain how you think some of what you have learnt in CS1010S will be helpful for you for the rest of your life and/or studies at NUS; or (b) tell us an interesting story about your experience with CS1010S this semester. [4 marks]

— E N D O F P A P E R —

Appendix

Parts of the Python documentation is given here for your reference.

List Methods

- `list.append(x)` Add an item to the end of the list.
- `list.extend(iterable)` Extend the list by appending all the items from the iterable.
- `list.insert(i, x)` Insert an item at a given position.
- `list.remove(x)` Remove the first item from the list whose value is *x*. It is an error if there is no such item.
- `list.pop([i])` Remove the item at the given position in the list, and return it. If no index is specified, removes and returns the last item in the list.
- `list.clear()` Remove all items from the list
- `list.index(x)` Return zero-based index in the list of the first item whose value is *x*. Raises a `ValueError` if there is no such item.
- `list.count(x)` Return the number of times x appears in the list.
- `list.sort(key=None, reverse=False)` Sort the items of the list in place.
- `list.reverse()` Reverse the elements of the list in place.
- `list.copy()` Return a shallow copy of the list.

Dictionary Methods

- `dict.clear()` Remove all items from the dictionary.
- `dict.copy()` Return a shallow copy of the dictionary.
- `dict.items()` Return a new view of the dictionary's items (`(key, value)` pairs).
- `dict.keys()` Return a new view of the dictionary's keys.
- `dict.pop(key[, default])` If *key* is in the dictionary, remove it and return its value, else return $default$. If $default$ is not given and *key* is not in the dictionary, a `KeyError` is raised.
- `dict.update([other])` Update the dictionary with the key/value pairs from *other*, overwriting existing keys. Return `None`.
- `dict.values()` Return a new view of the dictionary's values.

Scratch Paper

— H A P P Y H O L I D A Y S ! —

CS1010S — Programming Methodology
School of Computing
National University of Singapore

Final Assessment — Answer Sheet

2018/2019 Semester 2

Time allowed: 2 hours

Student No:	A								

Instructions (please read carefully):

1. Write down your **student number** on this answer sheet. DO NOT WRITE YOUR NAME!
2. This answer booklet comprises **TWELVE (12) pages**, including this cover page.
3. All questions must be answered in the space provided; no extra sheets will be accepted as answers. You may use the extra page behind this cover page if you need more space for your answers.
4. You must submit only the **ANSWER SHEET** and no other documents. The question set may be used as scratch paper.
5. An excerpt of the question may be provided to aid you in answering in the correct box. It is not the exact question. You should still refer to the original question in the question booklet.
6. You are allowed to use pencils, ball-pens or fountain pens, as you like as long as it is legible (no red color, please).
7. **Marks may be deducted** for i) unrecognisable handwriting, and/or ii) excessively long code. A general guide would be not more than twice the length of our model answers.

For Examiner's Use Only

Question	Marks
Q1	/ 30
Q2	/ 31
Q3	/ 19
Q4	/ 16
Q5	/ 4
Total	/100

This page is intentionally left blank.
Use it ONLY if you need extra space for your answers, and indicate the **question number clearly** as well as in the original answer box. **Do NOT** use it for your rough work.

Question 1A

[5 marks]

```
s = 'CS1010S-S0-C001'
d = {}
for i in range(len(s)):
    d[s[i]] = s[i%5]
print(d)
```

Question 1B

[5 marks]

```
d = {1:2, 2:4, 3:6, 4:8,
     5:0, 6:2, 7:4, 8:6,
     9:8, 0:3}
n = 1
for i in range(5):
    n = d[n]
print(n)
```

Question 1C

[5 marks]

```
a = [1, 2]
a += [a]
b = a.copy()
a[2] = 0
print(a)
print(b)
```

Question 1D

[5 marks]

```
def boo(x):
    try:
        i, j = int(x)
        print(i / j)
    except ZeroDivisionError:
        print('Why zero?')
    except ValueError:
        print('Wrong Value!')
    except IndexError:
        print('Wrong Index!')
    except TypeError:
        print('Wrong Type!')
boo(['ten', 0])
```

Question 1E

[5 marks]

```
def g(x):
    if x:
        return (x[-1], g(x[1:-1]), x[0])
    else:
        return ()
print( g([1, 2, 3, 4, 5]) )
```

Question 1F

[5 marks]

```
def bar(f, g):
    return lambda x: (lambda y: f(x))(g(x))
print( bar(lambda x:x+1, lambda x:x*2)(5) )
```

Question 2A Do you agree that lists should be used instead of tuples? [3 marks]

Question 2B [5 marks]

```
def step1(tet):
```

Coordinates are **List / Tuple** (circle one)

Question 2C [5 marks]

```
def step2(tet):
```

Question 2D Draw the box-pointer diagram of `s` and `t`. [4 marks]

Question 2E Explain why `rotate` does not give the right behaviour. [3 marks]

Question 2F [5 marks]

```
def can_place(tet, grid, row, col):
```

Question 2G

[6 marks]

```
def drop(tet, grid, col):
```

Question 3A

[5 marks]

```
def is_descendent(p_dict, anc, dec):
```

Question 3B [5 marks]

```
def convert(p_dict):
```

Question 3C [6 marks]

```
def is_descendent(c_dict, anc, dec):
```

Question 3D How will cousins having children affect our dictionaries? [3 marks]

Question 4A [3 marks]

```
>>> ipod = IPod(32)
>>> ipod.available_storage




>>> ipod.play_song("Somewhere Out There")




>>> ipod.call(65169121)
```

Question 4B In which class should the function `store_music` be placed? [2 marks]

Question 4C

[5 marks]

```
def store_song(self, song, size):
```

Question 4D

[6 marks]

```
class IPhone
```

Question 5 42 and the Meaning of Life [4 marks]

This page is intentionally left blank.
Use it ONLY if you need extra space for your answers, and indicate the **question number clearly** as well as in the original answer box. **Do NOT** use it for your rough work.

— END OF ANSWER SHEET —

Question 1A

[5 marks]

```
s = 'CS1010S-S0-C001'
d = {}
for i in range(len(s)):
    d[s[i]] = s[i%5]
print(d)
```

```
{'C': 'S', 'S': '0', '1': '1',
 '0': '0', '-': 'C'}
```

Test dictionary manipulation. Student should note that dictionary keys are unique and will be overwritten with subsequent assignments.

Question 1B

[5 marks]

```
d = {1:2, 2:4, 3:6, 4:8,
     5:0, 6:2, 7:4, 8:6,
     9:8, 0:3}
n = 1
for i in range(5):
    n = d[n]
print(n)
```

```
2
```

Test simply for-loop and matching keys in a dictionary. Basically starting at 1, the value shows the next key to read. Repeat 5 times. Students who count wrongly will end up at 6, or 4.

Question 1C

[5 marks]

```
a = [1, 2]
a += [a]
b = a.copy()
a[2] = 0
print(a)
print(b)
```

```
[1, 2, 0]
[1, 2, [1, 2, 0]]
```

Test if student understands list aliasing and that list copy is shallow.

Question 1D

[5 marks]

```
def boo(x):
    try:
        i, j = int(x)
        print(i / j)
    except ZeroDivisionError:
        print('Why zero?')
    except ValueError:
        print('Wrong Value!')
    except IndexError:
        print('Wrong Index!')
    except TypeError:
        print('Wrong Type!')
boo(['ten', 0])
```

```
'Wrong Type!'
```

Test if student know how to match the expected ErrorType and able to trace the exception handling

Question 1E

[5 marks]

```
def g(x):
    if x:
        return (x[-1], g(x[1:-1]), x[0])
    else:
        return ()
print( g([1, 2, 3, 4, 5]) )
```

```
(5, (4, (3, (), 3), 2) 1)
```

Test understanding of scoping of mutable structures and that slicing creates a new object.

Question 1F

[5 marks]

```
def bar(f, g):
    return lambda x: (lambda y: f(x))(g(x))
print( bar(lambda x:x+1, lambda x:x*2)(5) )
```

```
6
```

lambda substitution is as follow:

$$\texttt{bar(}\boxed{\lambda\ \texttt{x:x+1}}\texttt{,}\ \boxed{\lambda\ \texttt{x:x*2}}\texttt{)(5)}$$

$$\boxed{\lambda\ \texttt{x:}\ \boxed{\lambda\ \texttt{y:}\ \ \boxed{\lambda\ \texttt{x:x+1}}\texttt{(x)}}\texttt{(}\boxed{\lambda\ \texttt{x:x*2}}\texttt{(x))}}\texttt{(5)}$$

$$\boxed{\lambda\ \texttt{y:}\ \ \boxed{\lambda\ \texttt{x:x+1}}\texttt{(5)}}\texttt{(}\boxed{\lambda\ \texttt{x:x*2}}\texttt{(5))}$$

$$\boxed{\lambda\ \texttt{y:}\ \ \boxed{\lambda\ \texttt{x:x+1}}\texttt{(5)(10)}}$$

$$\boxed{\lambda\ \texttt{x:x+1}}\texttt{(5)}$$

6

Question 2A Do you agree that lists should be used instead of tuples? [3 marks]

No, because there is no need to mutate each coordinate tuple. We only have to mutate the list, and can change the elements of the list to new tuples.

Question 2B [5 marks]

Coordinates are **List / Tuple** (circle one)

```
def step1(tet):
    for i in range(len(tet)):
        tet[i] = (-tet[i][1], tet[i][0])

# Coordinates are list
def step1(tet):
    for t in tet:
        t[0], t[1] = -t[1], t[0]
```

Question 2C [5 marks]

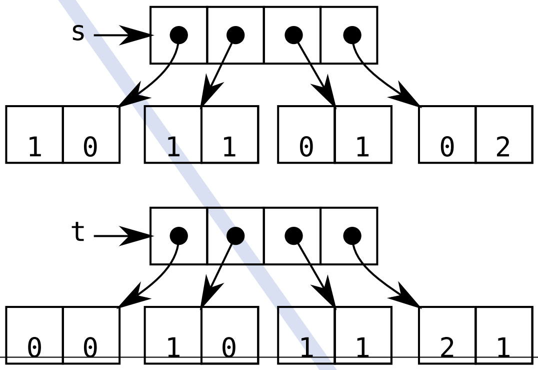
```
def step2(tet):
    # Note that only the x value may become negative
    # find the minimum x value
    min_x = min(tet)[0]  # since first element is the key

    # for tuples representation
    for i in range(len(tet)):
        tet[i] = (tet[i][0]-min_x, tet[i][1])


    ##########################
    # for lists representation
    for t in tet:
        t[0] -= min_x
```

Question 2D Draw the box-pointer diagram of `s` and `t`. [4 marks]

Answer depends on the implementation. If tuples are used or new points are created, then s will contain different point objects. Otherwise if implementation modifies existing points, then t will reference the same objects.



Question 2E Explain why `rotate` does not give the right behaviour. [3 marks]

`compose` applies g before f, so by right it should be called with `(step2, step1)`, but that is not sufficient to get it working.

That is because the functions do not return the modified tetromino in order to be chained/-composed. Instead `None` is returned, thus the second function is called with `None` as input, leading to error.

Question 2F [5 marks]

```
def can_place(tet, grid, row, col):
    for x, y in tet:
        if 0 >= row+y >= len(grid):  # check if exceed grid vertically
            return False
        if grid[row+y][col+x]:       # check existing overlap
            return False
    return True
```

-1 Checking `grid[row]][col]`. It is not always the case that this square is in the tetromino.
-1 Fail to check if part of the tetromino is placed outside the grid. Only need to check vertically as question states it will not exceed the width. This is needed to simplify the answer to the next part.
-1 Check for bounds have to be before check of overlap, otherwise `IndexError` will occur. One easy way is to avoid this complication is to just use `try` and return `False` on `except`.
-1 `tet` is modified.

Question 2G [6 marks]

```
def drop(tet, grid, col):
    row = 0
    while can_place(tet, grid, row, col):
        row += 1
    if row == 0:  # cannot place at top row
        return false
    # update the grid
    for x,y in tet:
        grid[row-1+y][col+x] = True
    return True
```

Common mistakes:

-1 did not check placing at top (row 0)
-1 did not check placing at bottom (row `len(grid)`)
-2 search from bottom-up instead of top-down

Question 3A [5 marks]

```
def is_descendent(p_dict, anc, dec):
    if dec not in p_dict:
        return False
    if p_dict[dec] == anc:
        return True
    else:
        return is_descendent(p_dict, anc, p_dict[dec])
```

Question 3B [5 marks]

```python
def convert(p_dict):
    d = {}
    for child, parent in p_dict.items():
        if parent not in d:
            d[parent] = []
        if child not in d:
            d[child] = []
        d[parent].append(child)
    return d
```

Question 3C [6 marks]

```python
def is_descendent(c_dict, anc, dec):
    if dec in c_dict[anc]:
        return True
    else:
        for child in c_dict[anc]:
            if is_descendent(c_dict, child, dec):
                return True
        return False
```

Question 3D How will cousins having children affect our dictionaries? [3 marks]

There is no issue with the *parent→children* dictionary, just that both Freddy and Gerene will have the same children in their value list.

But for the *child→parent* dictionary, the problem is the values are just a single string, so each person can only have one parent. Thus, it is not possible to capture the relationship of two parents in the dictionary.

Question 4A [3 marks]

```
>>> ipod = IPod(32)
>>> ipod.available_storage
bound method IPod.available storage
```

Indicating that it will return a function or method will suffice

```
>>> ipod.play_song("Somewhere Out There")
Somewhere Out There not found
```

```
>>> ipod.call(65169121)
Error: ipod does not have attribute call
```

Question 4B In which class should the function `store_music` be placed? [2 marks]

+2 `MusicPlayer`, is the best place as it is the class managing songs.
+1 Placing it in `StorageDevice` will still work but it is not the best place as `StorageDevice` does not know anything about songs.

Question 4C

[5 marks]

```
def store_song(self, song, size):
    if size+1 > self.available_storage():
        print("Not enough Memory")
    else:
        self.items.append(song)
        self.memory += size
        print(song, "stored")
        print(self.available_storage(), "memory left")
```

Question 4D

[6 marks]

```
class IPhone(PhoneDevice, MusicPlayer):  # order is important
    def __init__(self, memory):  # iPhone has no initial phone number
        super().__init__(0, memory)

    # alternatively we can init with phone number and arbitary memory
    # def __init__(self, number)
    #     super().__init__(number, 0)

    def search(self, text):
        print("Searching", text)
```

Question 5 42 and the Meaning of Life [4 marks]

The student will be awarded points as long as he/she is coherent and doesn't say something obviously wrong.