CS 88 Spring 2022 Computational Structures in Data Science

FINAL

INSTRUCTIONS

- **Do NOT open the exam until you are instructed to do so!**
- You **must not** collaborate with anyone inside or outside of CS88.
- You **must not** use any internet resources to answer the questions.
- If you are taking an online exam, at this point you should have started your Zoom / screen recording. If something happens during the exam, focus on the exam! Do not spend more than a few minutes dealing with proctoring.
- When a question specifies that you must rewrite the completed function, you should **not** recopy the doctests.
- The exam is closed book, closed computer, closed calculator, except your hand-written 8.5" x 11" cheat sheets of your own creation and the official CS88 Reference Sheet

Full Name	
Student ID Number	
Official Berkeley Email (@berkeley.edu)	`CS88 In Person`
What room are you in?	
Name of the person to your left	
Name of the person to your right	
By my signature, I certify that all the work on this exam is my own, and I will not discuss it with anyone until exam session is over. **(please sign)**	

POLICIES & CLARIFICATIONS

- If you need to use the restroom, bring your phone and exam to the front of the room.
- For fill-in-the-blank coding problems, we will only grade work written in the provided blanks.
- Unless otherwise specified, you are allowed to reference functions defined in previous parts of the same question.
    - **Online Exams: You may start you exam as soon as you are given the password.**
    - **You may have a digitial version of the CS88 Reference Sheet, and the corrects doc, but no other files. You notes should be handwritten**
    - Remember to view the Corrections Document on the board. (Online students, open the link)

1. (10.0 points) Conceptual Questions

(a) (2.0 pt) Suppose the following code has been executed:

```
x = [1, 2, 3]
y = [4, 5]
```

Which 2 choices below will produce the same result?

- [x] `x = x + y`
- [ ] `x.append(y)`
- [x] `x.extend(y)`

(b) (2.0 pt) Kevin is trying to implement a cake ADT and a party ADT:

```
# Cake ADT
def make_cake(flavor, slices):
    return [flavor, slices]

# Party ADT
def make_party(guests, cake):
    return {'guests': guests, 'cake': cake}
```

Kevin is trying to implement the `finish_cake` method **in the party ADT**, which finishes all of the cake slices in the cake ADT. Does the following implementation violate any abstraction barriers?

```
def finish_cake(party):
    cake = party['cake']
    cake[1] = 0
```

- (●) Yes, it violates the abstraction barrier for the cake ADT
- ( ) Yes, it violates the abstraction barrier for the party ADT
- ( ) No, it does not violate any abstraction barriers

(c) (2.0 pt) Kevin is trying to implement the `get_flavor` method **in the cake ADT**, which returns the flavor of the cake ADT. Does the following implementation violate any abstraction barriers?

```
def get_flavor(cake):
    return cake[0]
```

- ( ) Yes, it violates the abstraction barrier for the cake ADT
- ( ) Yes, it violates the abstraction barrier for the party ADT
- (●) No, it does not violate any abstraction barriers

(d) (2.0 pt) In our linked list class, what is the *BEST* practice for determining whether we are at the end of a linked list?

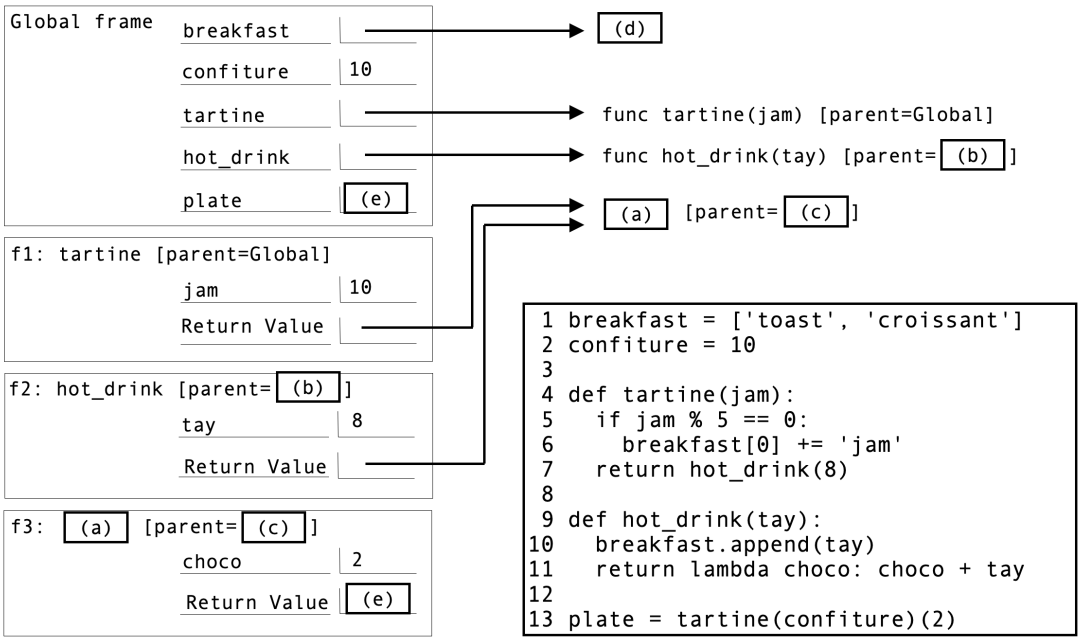
- (●) Compare the current linked list node to `Link.empty`
- ( ) Compare the current linked list node to `()`
- ( ) Find when the code causes an attribute error.

(e) (2.0 pt) All generators are iterators but not all iterators are generators.

- (●) True
- ( ) False

2. (10.0 points) Breakfast Time!

Fill in the blanks to complete the environment diagram. All the code used is in the box to the right, and the code runs to completion with no errors. Some arrows have been removed from the diagram. You may wish to draw in those arrows, but it is not required.



(a) (2.0 pt) What is the return value of the `f1` frame?

- ○ `func hot_drink`
- ● `func lambda`
- ○ `func tartine`

(b) (2.0 pt) What is the parent frame of the `hot_drink` function in the `f2` frame?

- ● Global
- ○ f1

(c) (2.0 pt) What is the parent of the `lambda` function in the `f3` frame?

- ○ Global
- ○ f1
- ● f2

(d) (2.0 pt) What is the value of `breakfast` in the Global frame when the environment diagram is complete?

['toastjam', 'croissant', 8]

(e) (2.0 pt) What is the value of `plate` in the Global frame when the environment diagram is complete?

10

3. (10.0 points) What Would Python Do (WWPD)

For each expression below, write the output displayed by the interactive Python interpreter when the expression is evaluated. The output may have multiple lines. If an error occurs, write "Error" (if any lines are displayed before the error, include those in your output). If a function is returned, write "Function".

```
f1 = lambda x: lambda y: (x * x) + (y * y)

def f2(lnk):
    idx = 0
    while lnk != Link.empty:
        print(lnk.first[idx])
        idx += 1
        lnk = lnk.rest

class MessengerAccount():
    total_accounts = 0
    def __init__(self, owner):
        self.owner = owner
        self.notification = 0
        self.inbox = []
        MessengerAccount.total_accounts += 1

    def send_msg(self, recipient, msg):
        recipient.receive_msg(self.owner, msg)

    def receive_msg(self, sender, msg):
        self.notification += 1
        self.inbox.append((sender, msg))

    def read_msg(self):
        for _ in range(len(self.inbox)):
            msg = self.inbox.pop()
            print("From " + msg[0] + ": " + msg[1])
```

(a) (2.0 pt) `>>> f1(4)(3)`

```
25
```

(b) (2.0 pt) `>>> f2(Link("horse", Link("koala", Link("puppy", Link("eel")))))`

```
h
o
p
Error
```

(c) (2.0 pt)

```
>>> chi = MessengerAccount("Chi")
>>> lukas = MessengerAccount("Lukas")
>>> lukas.total_accounts
```

```
2
```

(d) (2.0 pt)

```
>>> lukas.send_msg(chi, "Check out my Youtube channel!")
>>> lukas.send_msg(chi, "Like and sub (:")
>>> chi.read_msg()
```

```
From Lukas:  Like and sub (:
From Lukas:  Check out my Youtube channel!
```

(e) (2.0 pt)

```
>>> chi.send_msg(lukas, "Cool!")
>>> print(lukas.notification, chi.notification)
```

```
1 2
```

4. (6.0 points) Decider Mapping

Complete the function `decider_map`. `decider_map` takes in a list `lst` and a function `decider`. `decider` is a higher order function that returns a function based on two inputs: a `value` and the `index` of that value in `lst`. For every `value` in `lst`, determine what function needs to be applied to that `value` using `decider`, and return a new list where all values in `lst` have the function chosen by `decider` applied to them.

```
def decider_map(lst, decider):
    """
    >>> def decider(index, value):
    ...   if index + value >= 0:
    ...     return lambda x : x * 2
    ...   else:
    ...     return lambda x : -x
    >>> decider_mapping([], decider)
    []
    >>> decider_mapping([-1, -1], decider)
    [1, -2]
    >>> decider_mapping([1, -5, -2, 4], decider)
    [2, 5, -4, 8]
    """
    output = []
    for i in ____________________:
        decided_fn = ____________________
        output.____________(__________________)
    return output
```

(a) (6.0 pt)

```
def decider_map(lst, decider):
    output = []
    for i in range(len(lst)):
        decided_fn = decider(i, lst[i])
        output.append(decided_fn(lst[i]))
         Alt:  output.extend([decided_fn(lst[i])])
    return output
```

5. (6.0 points) Cascading Numbers

Complete the function `cascade`, which takes in an integer `base`, a function `fn`, and a non-negative integer `count`. `cascade` returns a sequence of numbers starting with `base`, `fn(base)`, `fn(fn(base))`, ... and so on, `count` number of times and then continues the sequence in reverse back to `base`.

```
def cascade(base, fn, count):
    """
    >>> cascade(5, lambda x: 2 * x, 0)
    []
    >>> cascade(5, lambda x: 2 * x, 1)
    [5]
    >>> cascade(5, lambda x: 2 * x, 2)
    [5, 10, 5]
    >>> cascade(100, lambda x: x - 4, 3)
    [100, 96, 92, 96, 100]
    >>> cascade(4, lambda x: x - 4, 5)
    [100, 96, 92, 88, 84, 88, 92, 96, 100]
    """
    if count == 0:
        return _______________________________
    elif count == 1:
        return _______________________________
    else:
        middle = cascade(_____________________________)
        return _______________________________
```

(a) (5.0 pt)

```
def cascade(base, fn, count):
    if count == 0:
        return []
    elif count == 1:
        return [base]
    else:
        middle = cascade(fn(base), fn, count - 1)
        return [base] + middle + [base]
```

(b) With respect to the term `count`, what is the runtime of the function `cascade`?

- ○ Constant
- ○ Logarithmic
- ● Linear
- ○ Polynomial
- ○ Exponential

6. (4.0 points) Median Scores

Complete the function `median_score` which takes in a dictionary `student_scores` that maps each test score to the list of students who earned that score, and returns the median score on the test. You may use the provided `median` function which takes in an unordered list of integers and returns its median.

The median is the middle value in an odd-length ordered list (see the first doctest) or the average of the middle two values in an even-length ordered list (see the second doctest).

```
def median(lst):
    """
    >>> median([1, 6, 7])
    6
    >>> median([1, 2, 8, 6])
    4.0
    """
    # Implementation not shown

def median_score(student_scores):
    """
    >>> d = {3 : ["bob", "sally"], 4 : ["Arun", "Lars", "Ken"]}
    >>> median_score(d)
    4
    >>> d = {3 : ["bob", "sally"], 4 : ["Arun", "Lars"]}
    >>> median_score(d)
    3.5
    """
    all_scores = []
    for score in student_scores:
        students = student_scores[score]
        for ____________________________________________________:
            ____________________________________________________
    return ____________________________________________________
```

(a) (4.0 pt)

```
def median_score(student_scores):
    all_scores = []
    for score in student_scores:
        students = student_scores[score]
        for _ in range(len(students)):
            all_scores.append(score)
    return median(all_scores)
```

7. (8.0 points) Wordle

You are writing a clone of the hit game Wordle!

Implement the `check_correctness` function which takes in a linked list, `guess` (with each element as an individual letter) and a string, `correct_word`. The function should output a linked list with one of the following colors at each of the corresponding positions in `guess`:

- `'green'`: If letter in `guess` is both in the `correct_word` and in the correct position
- `'yellow'`: If letter in `guess` is in the `correct_word` but not in the correct position
- `'black'`: If letter in `guess` is not in `correct_word`

Both `guess` and `correct_word` will always be the same length, and you can assume they have unique and lowercase letters.

Note: You can use indexing (`[]`) and the `in` operator on strings, just like lists! (But you *cannot* use them on a `Link` object.)

```
def check_correctness(guess, correct_word):
    '''
    >>> guess = Link('c', Link('r', Link('a', Link('n', Link('e')))))
    >>> correct_word = "train"
    >>> check_correctness(guess, correct_word)
    Link('black', Link('green', Link('green', Link('yellow', Link('black')))))
    '''
    def helper(guess, correct_word, i):
        if guess is Link.empty:
            return _________________________
        rest = ___________________________________
        if ______________________:
            return Link(________________, rest)
        elif ______________________:
            return Link(________________, rest)
        else:
            return Link(________________, rest)
    return helper(guess, correct_word, 0)
```

(a) (8.0 pt)

```
def check_correctness(guess, correct_word):
    def helper(guess, correct_word, i):
        if guess is Link.empty:
            return guess
        rest = helper(guess.rest, correct_word, i+1)
        if guess.first == correct_word[i]:
            return Link('green', rest)
        elif guess.first in correct_word:
            return Link('yellow', rest)
        else:
            return Link('black', rest)
    return helper(guess, correct_word, 0)
```

8. (6.0 points) Maze

You are simulating a game where a player is walking through a maze by choosing right, `'R'`, or left, `'L'`, repeatedly in a sequence.

Complete the generator `checker`, which takes in `correct_gen`, a generator that yields the next step the player *should* take, and `guess_gen`, a generator that yields the next step the player CHOOSES to take.

`checker` simulates the game by yielding `'Ok'` if the player chooses the correct next step and `'Lose'` otherwise. If the player completes the maze correctly, `checker` should yield `'Maze Complete'`.

Assume that `correct_gen` and `guess_gen` will yield the same number of elements.

```
def checker(correct_gen, guess_gen):
    """
    >>> def path_gen(lst):
    ...     for el in lst:
    ...         yield el
    ...
    >>> correct_path = path_gen(['R', 'L', 'R'])
    >>> guess_path = path_gen(['R', 'R', 'R'])
    >>> checker_gen = checker(correct_path, guess_path) # Simulate Losing
    >>> next(checker_gen)
    'Ok'
    >>> next(checker_gen)
    'Lose'
    >>> checker_gen = checker(path_gen(['R', 'L']), path_gen(['R', 'L'])) # Simulate Winning
    >>> next(checker_gen)
    'Ok'
    >>> next(checker_gen)
    'Ok'
    >>> next(checker_gen)
    'Maze Complete'
    """
    while True:
        try:
            correct_step = ____________________________
            guess_step = ____________________________
            if correct_step != guess_step:
                ____________________________
                return
            else:
                ____________________________
        except StopIteration:
            ____________________________
```

(a) (6.0 pt)

```
def checker(correct_gen, guess_gen):
    while True:
        try:
            correct_step = next(correct_gen)
            guess_step = next(guess_gen)
            if correct_step != guess_step:
                yield 'Lose'
                return
            else:
                yield 'Ok'
        except StopIteration:
            yield 'Maze Complete'
```

9. (10.0 points) Color Schemes

You will be implementing a system for creating color palettes, using two classes `Color` and `Color Palette`.

To start, you will finish the class `Color`. The `Color` class has three instance attributes, `r`, `g`, and `b`, which stores the RGB values of the color. The RGB color model is a common way to encode colors.

```
class Color:
    def __init__(self, red, green, blue):
        self.r = red
        self.g = green
        self.b = blue

    def dissimilarity(self, other):
        r_diff = ________________________________________
        g_diff = ________________________________________
        b_diff = ________________________________________
        ________________________________________________

     def __repr__(self): #you can ignore this!!!
         return f"Color({self.r}, {self.g}, {self.b})"
```

(a) (3.0 pt) All that's left is to implement the method `dissimilarity`, which computes the dissimilarity between two colors. Dissimilarity is equal to the sum of the absolute differences between the `r`, `g`, and `b` attributes of the two colors.

```
def dissimilarity(self, other):
    """
    >>> blue = Color(0, 0, 100)
    >>> green = Color(0, 100, 0)
    >>> blue.dissimilarity(green) # abs(0-0) + abs(0-100) + abs(100-0)
    200
    """
    r_diff = ________________________________________
    g_diff = ________________________________________
    b_diff = ________________________________________
    ________________________________________________
```

```
def dissimilarity(self, other):
    r_diff = abs(self.r - other.r)
    g_diff = abs(self.g - other.g)
    b_diff = abs(self.b - other.b)
    return r_diff + g_diff + b_diff
```

(b) (4.0 pt) Now, we're ready to work on the `ColorPalette` class. A instance of the `ColorPalette` class has two attributes: 1. `colors`: a list of instances of the `Color` class 2. `threshold`: a int indicating how similar two colors can be

You will be completing the `add_to_palette` method.

```
class ColorPalette:
    def __init__(self, color_lst, threshold):
        self.threshold = threshold
        self.colors = []
        for c in color_lst:
            self.add_to_palette(c)

    def add_to_palette(self, color):
        for c in self.colors:
            if ________________________________________:
                ________________________________________________
                ________________________________________________
        ____________________________________________
```

Implement the method `add_to_palette` which attempts to add the color `color` to a ColorPalette.

We do not want to add a Color to a ColorPalette if its dissimilarity to any Color in the `colors` list is less than the `threshold` instance attribute. You may not need all the lines of code provided. Assume that the `Color` class is implemented correctly.

If the Color is too similar to the colors in the palette, print "Too Similar!" (see doctests).

```
def add_to_palette(self, color):
    """
    >>> blue = Color(0, 0, 100)
    >>> the_blues = ColorPalette([blue], 5)
    >>> the_blues.colors
    [Color(0, 0, 100)]
    >>> blue_again = Color(0, 0, 97)
    >>> the_blues.add_to_palette(blue_again)
    Too similar!
    >>> the_blues.colors
    [Color(0, 0, 100)]
    >>> green = Color(0, 100, 0)
    >>> the_blues.add_to_palette(green)
    >>> the_blues.colors
    [Color(0, 0, 100), Color(0, 100, 0)]
    """
    for c in self.colors:
        if ________________________________________:
            ________________________________________________
            ________________________________________________
    ____________________________________________
```

```
def add_to_palette(self, color):
    for c in self.colors:
        if color.dissimilarity(c) < self.threshold:
            print("Too similar!")
            return
    self.colors += [color]
```

(c) (3.0 pt) You decide to implement a color palette called `TriColor`.

This palette accepts two colors, `c1` and `c2`, and an int `threshold` as input. It computes a third color which is the average of the two input colors. These three colors should all form one color palette, which has a similarity threshold equal to `threshold`.

Implement the constructor for `TriColor`. You may not need all the lines of code provided. Assume that the `Color` and `ColorPalette` classes are implemented correctly.

```
class TriColor(____________________):
    def __init__(self, c1, c2, threshold):
        """
        >>> blue = Color(0.0, 0.0, 100.0)
        >>> green = Color(0.0, 100.0, 0.0)
        >>> blue_green = TriColor(blue, green, 10)
        >>> blue_green.colors
        [Color(0.0, 0.0, 100.0), Color(0.0, 100.0, 0.0), Color(0.0, 50.0, 50.0)]
        >>> the_blues = TriColor(blue, blue, 10)
        Too similar!
        Too similar!
        >>> the_blues.colors
        [Color(0.0, 0.0, 100.0)]
        """
        avg_r = (c1.r + c2.r)/2
        avg_g = (c1.g + c2.g)/2
        avg_b = (c1.b + c2.b)/2
        ------------------------------------------------
        ------------------------------------------------
```

```
class TriColor(ColorPalette):
    def __init__(self, c1, c2, threshold):
        avg_r = (c1.r + c2.r)/2
        avg_g = (c1.g + c2.g)/2
        avg_b = (c1.b + c2.b)/2
        avg_color = Color(avg_r, avg_g, avg_b)
        super().__init__([c1, c2, avg_color], threshold)
```

10. (6.0 points) Two Tree

A Two Tree is a Tree where every node has 1. A value of either 0 or 1. 2. At most two children.

An example of a valid Two Tree is `Tree(0, [Tree(1), Tree(1)])`. Examples of invalid Two Trees include `Tree(2)` and `Tree(1, [Tree(0), Tree(0), Tree(1)])`.

The (possibly buggy!) `add_nums` function should return the sum of all the values in the input `t`, which is a Two Tree. For each (a)-(d), write any valid Two Tree `t` that makes the statement true. If no such valid Two Tree exists, write "Impossible".

```
def add_nums(t):
    count = t.value
    for b in t.branches:
        count += b.value
    return count
```

(a) (1.5 pt) `add_nums(t)` is supposed to return 2, and does return 2.

```
Tree(1, [Tree(1)])
```

(b) (1.5 pt) `add_nums(t)` is supposed to return 3, and does return 3.

```
Tree(1, [Tree(1), Tree(1)])
```

(c) (1.5 pt) `add_nums(t)` is supposed to return 3, and does NOT return 3.

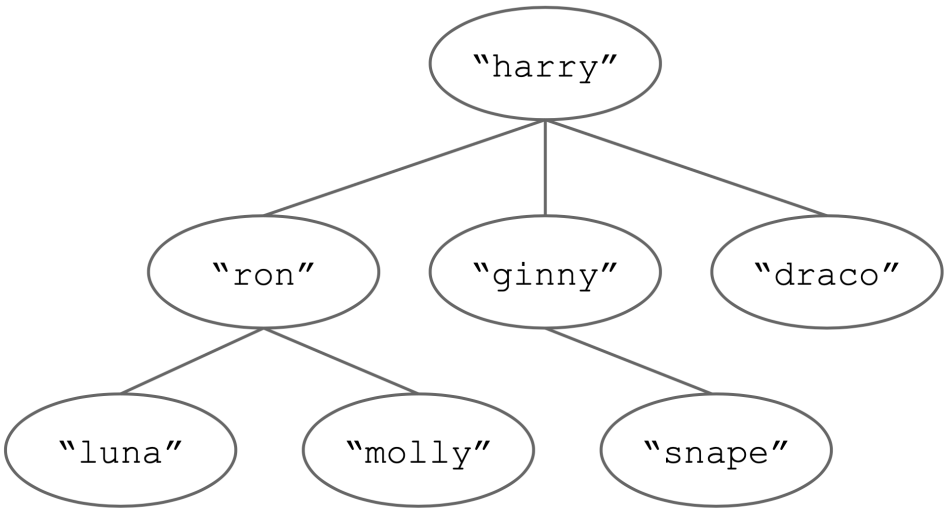
```
Tree(1, [Tree(0, [Tree(1)]), Tree(1)])
```

(d) (1.5 pt) `add_nums(t)` is supposed to return 4, and does return 4.

Impossible!

11. (6.0 points) SorcerTree

CS 88 is getting popular as a course, and more and more people are recommending it to their friends! We are representing these recommendations as a tree of students (represented as strings), where the branches of a student `s` are the students that `s` recommended to take the course. For example, in the below tree, Harry recommended Ron, Ginny, and Draco to take the class.



Given a tree `t` and a list `star_students`, return a list of two element tuples, where each tuple has a student from `star_students` and the person that recommended them to take CS 88. Assume that every student in the `star_students` list appears in the tree `t` once.

```
def find_recommenders(t, star_students):
    """
    >>> t = Tree("harry", [ Tree("ron", [Tree("luna"), Tree("molly")]),
        Tree("ginny", [Tree("snape")]), Tree("draco") ])
    >>> find_recommenders(t, ["ginny"])
    [('ginny', 'harry')]
    >>> find_recommenders(t, ["ron", "molly", "snape"])
    [('ron', 'harry'), ('molly', 'ron'), ('snape', 'ginny')]
    >>> find_recommenders(t, ["harry", "luna"])
    [('luna', 'ron')]    # No one recommended harry to take the class, as harry is the root node
    """
    result = []
    for b in t.branches:
        if _____________________________________:
            result += _____________________________________
        result += _____________________________________
    return result
```

(a) (6.0 pt)

```
def find_recommenders(t, star_students):
    result = []
    for b in t.branches:
        if b.value in star_students:
            result += [(b.value, t.value)]
        result += find_recommenders(b, star_students)
    return result
```

12. (10.0 points) SQLiving Spaces

Answer the following questions given tables of the following form.

student

name	grade	res_hall
Sebastian	freshman	Triumph
Karim	sophomore	Triumph
Jessica	sophomore	Oasis
Hetal	junior	Oasis
Amit	junior	Empire
Tommy	junior	Dynamic
Lukas	junior	Empire
Anjali	junior	Millenium
Matt	junior	Millenium
Shreya	senior	Oasis
Kevin	senior	Millenium
Chi	senior	Dynamic
Minnie	senior	Dynamic

building

res_hall	complex	year
Triumph	Unit A	1980
Empire	Unit A	1976
Millenium	Unit B	1983
Oasis	Unit C	1964
Dynamic	Unit B	1976
Pinnacle	Unit A	1971

(a) (3.0 pt) Write a SQL query that retrieves the `name` and `res_hall` of all **sophomore** students. The expected output is given below.

name	res_hall
Karim	Triumph
Jessica	Oasis

```
SELECT name, res_hall
FROM student
WHERE grade = 'sophomore'
```

(b) (3.0 pt) Write a SQL query that retrieves all rows corresponding to the buildings in **Unit A** in **alphabetical order** by building name. The expected output is given below.

res_hall	complex	year
Empire	Unit A	1976
Pinnacle	Unit A	1971
Triumph	Unit A	1980

```
SELECT * FROM building
WHERE complex = 'Unit A'
ORDER BY res_hall
```

(c) (4.0 pt) Write a SQL query that retrieves the **total** number of **junior** students who live in each complex as `total`. The expected output is given below.

complex	total
Unit A	2
Unit B	3
Unit C	1

```
SELECT complex, COUNT(*) AS total
FROM student AS s, building AS b
WHERE s.res_hall = b.res_hall
AND grade = 'junior'
GROUP BY complex
```

No more questions.