ICS 31 • UC IRVINE
FALL 2015 • DAVID G. KAY
****
****** KEY ******
****

YOUR NAME ______________________________
YOUR STUDENT ID (8 DIGITS) ________________
YOUR UCINET ID ___________________________

YOUR LAB SECTION (CIRCLE ONE):

1.	8:00a	Prateek B. Swamy
2.	10:00	Prateek B. Swamy
3.	12:00	Rohit Malhotra
4.	2:00	Rohit Malhotra
5.	4:00	Roeland Singer-Heinze
6.	6:00	Roeland Singer-Heinze
7.	8:00p	Akshat Amrish Patel
8.	8:00a	Shibani Konchady
9.	10:00	Shibani Konchady
10.	12:00	Vignesh Raghunathan
11.	2:00	Vignesh Raghunathan
12.	4:00	Akshat Amrish Patel
13.	8:00a	Kartic Saxena
14.	10:00	Kartic Saxena
15.	12:00	Archit Dey
16.	2:00	Archit Dey

First Midterm

You have 75 minutes (until the end of the class period) to complete this exam. There are 65 points possible, so allow approximately one minute per point and you'll have plenty of time left over.

Please read all the problems carefully. If you have a question on what a problem means or what it calls for, ask us. Unless a problem specifically asks about errors, you should assume that each problem is correct and solvable; ask us if you believe otherwise.

In answering these questions, you may use any Python 3 features we have covered in class, in the text, in the lab assignments, or earlier on the exam, unless a problem says otherwise. Use more advanced features at your own risk; you must use them correctly. If a question asks for a single item (e.g., one word, identifier, or constant), supplying more than one will probably not receive credit.

Remember, stay cool! If you run into trouble on a problem, go on to the next one. Later on, you can go back if you have time. Don't let yourself get stuck on any one problem.

You may not share with or receive from anyone besides the instructor or TAs any information or materials during the exam. You may not use any electronic devices.

Please write your answers clearly and neatly—we can't give you credit if we can't decipher what you've written.

We'll give partial credit for partially correct answers, so writing something is better than writing nothing. But be sure to answer just what the question asks.

Good luck!

Problem 1 (11 points)	
Problem 2 (9 points)6	
Problem 3 (19 points)	
Problem 4 (5 points)	
Problem 5 (6 points)	
Problem 6 (6 points)	
Problem 7 (9 points)	
Total (65 points)	

Problem 1 (11 points) **Topic: Simple expressions with numbers, lists, strings**

Use the following definitions in this problem: **# Note to students: The correct answers are as shown.**

```
THIS_YEAR = 2015                    # We gave credit in some places for variations in quotes or
a = "UCI's 50th"                    # capitalization, but that doesn't mean they're strictly correct
L = ['class', 'book', 'computer', 'dorm', 'roommate', 'professor', 'smartphone']
```

(a) (4 points) What does Python print as it executes the following sequence of statements? (Write your answers in the blank space to the right of the code.)

```
print("Go Eaters!")      # Go Eaters! [Also credit on this prob. if quotes] SCORING: 1/2 pt Version
print(THIS_YEAR * 10)    # 20150 [no quotes here] SCORING: 1/2pt
print(THIS_YEAR, "is", a)      # 2015 is UCI's 50th [SCORING:Total 2 pts: 1/2 for each item, 1/2 for UCI's]
print(len(L) >= 10)         # False [Credit on this problem for uncapitalized false or all-caps] 1 pt.
```

(b) (3 points) What does Python print as it executes the following sequence of statements? (Write your answers in the blank space to the right of the code.) Remember zero-based indexing. **SCORE 0.5/line**

```
print(a[1])                   # C [for this question, credit for 'C']
print(a[0] == 'I')            # False [capitalization variants as above, this problem]
print(a[2] + 'CE')            # ICE [also credit for 'ice', but note all incorrect capitalization]
print('500' in a)             # False [capitalization variants as above, this problem]
print(a in '50th')            # False [capitalization variants as above, this problem]
print(len('OK'*3), 'OK'*3)    # 6 OKOKOK [credit if one set of quotes around OKOKOK, otherwise no]
```

(c) (4 points) What does Python print as it executes the following sequence of statements? (Write your answers in the blank space to the right of the code.) Remember zero-based indexing. **SCORE 1pt/line**

```
print(L[-1], L[0])               # smartphone class [Look for ans. from the wrong version]
print(len(L), len(L[1:]))        # 7 6
print(L[1], L[3], L[5])          # book dorm professor [also credit for quotes around each item]
print("Dorm" in L)               # False [credit for false/FALSE, though the point here is that
                                 #    capitalization is significant.]
```

Problem 2 (9 points) **Topic: Defining and modifying namedtuples**

The Anteater Cafe represents each coffee drink in namedtuple called Drink that has four fields: the name of the drink, the size in ounces, customization notes (e.g., "decaf no-foam"), and its price.

(a) (3 points) Which of the following defines a namedtuple that satisfies this specification? Circle *one or more* of A, B, C, D, E, or F; more than one may be correct. **SCORING: –1 for each error (wrongly circled, wrongly blank)**

```
A. Coffee = namedtuple('Coffee', 'name ounces notes cost')
B. Drink = namedtuple('Drink', 'Caffe Latte', 12, 'decaf no-foam', 3.55)
C. Drink = namedtuple('Drink', 'Espresso, 3, none, 2.75')
D. Drink = namedtuple('Drink', 'title ounces customization price') # THIS
E. Drink = namedtuple('Drink', 'name size notes price')    # THIS ONE
F. Drink = Drink('Cappucino', 20, 'half-caf nonfat', 4.45)
```

(b) (3 points) Which of the following creates a new Drink object, following the definition above, to represent a 20-ounce iced decaf Americano that costs $4.75? Circle *one or more* of A, B, C, D, or E; more than one may be correct.

A. `new_drink = Drink.Americano(20, 'iced decaf', 4.75)`
B. `new_drink = Drink('Americano, 20, iced decaf, 4.75')`
C. `new_drink = Drink('Americano', 20, 'iced decaf', 4.75)` **# THIS ONE**
D. `new_drink = namedtuple('Drink', 'Americano 20 iced decaf 4.75')`
E. `new_drink = Drink(20, 'iced decaf', 'Americano', 4.75)`

(c) (3 points) You want to increase the price of each drink by fifteen cents. Which of the following statements correctly increases the price of the Drink object stored in `a_drink` (according to at least one of the correct definitions in part (a))? Circle *one or more* of A, B, C, D, E, or F; more than one may be correct.

A. `a_drink = a_drink._replace(price = price + 0.15)`
B. `a_drink.price = a_drink.price + 0.15`
C. `a_drink = Drink(a_drink.name, a_drink.size, a_drink.notes, a_drink.price + 0.15)` **#OK**
D. `a_drink = a_drink._replace(price = 4.90)`
E. `a_drink = Drink(price = a_drink.price + 15)`
F. `print(a_drink.price + 0.15)`

Problem 3 (19 points) **Topic: Functions, sorting a list of namedtuples**

The Anteater Cafe has branches worldwide. Their management has asked you to analyze a recent customer satisfaction survey. Customers at every branch have given the branch a 0–100 score in three categories: quality of the drinks, the service, and the ambience (physical environment). You use this namedtuple to represent a branch's survey results:

```
Branch = namedtuple('Branch', 'location ID drinks service ambience')
```

The location is a string, the ID is an int, and the remaining fields are floats storing the average rating from all customers on each category for that branch.

(a) (4 points) In the function definition below, fill in each blank with a single Python constant, operator, or identifier name (variable, function, attribute, method) to satisfy the problem specification.

```
def overall_satisfaction(b: Branch) -> float:
    """ Return the branch's overall satisfaction score, in the range 0-100, computed
        as an equally-weighted average of its drinks, service, and ambience scores.
    """
    return (__________.drinks __________ b.service + b.__________) / __________
```

return (b.drinks + b.service + b.ambience)/3 **SCORING: 1 point each**

(b) (1 point) Fill in each bank in the assert statements below so that each assertion will be true (assuming a correct solution to part (a)).

```
assert overall_satisfaction(Branch('Irvine', 111, 25, 50, 75)) == _____

assert overall_satisfaction(Branch('Costa Mesa', 222, _____, _____, _____)) == 100
```

First line: 50 (0.5 point) **Second line: 100 100 100 (0.5 point)**

(c) (6 points) What do the following statements print? [You can do the arithmetic in your head.]

```
B1 = Branch('Anaheim', 333, 85, 90, 95)
B2 = Branch('Laguna Beach', 444, 93, 93, 93)
print(B1.location, "(#", B1.ID, '):', overall_satisfaction(B1), 'overall/100')
print(B2.location, "(#", B2.ID, '):', overall_satisfaction(B2), 'overall/100')
```

Anaheim (# 333): 90 overall/100

Laguna Beach (# 444): 93 overall/100

SCORING: 1/2 point for each location, 1/2 point for each ID number (total 2)

1 point for each correct rating number with the right name (with or without .0) (total 2)

1 point for the correct parentheses, #, colon [partial credit okay; don't worry about extra blanks] (total 1)

1 point for everything else correct (without worrying about extra spacing) (total 1)

(d) (2 points) In the function definition below, fill in each blank with a single Python constant, operator, or identifier name (variable, function, attribute, method) to satisfy the problem specification.

```
def Branch_drinks(b: Branch) -> float:
    ''' Return a Branch's score for drinks
    '''
    return ___________ . ___________       # b   drinks
```

(e) (6 points) In addition to the function `Branch_drinks` defined above, assume you have also defined similar functions called `Branch_service` and `Branch_ambience` that return the values of those fields. Also, suppose that Anteater Cafe has a list of 750 Branches called `BL`

(e.1) Fill in the blanks below to reorder `BL` by each Branch's service score, lowest to highest:

```
________________.sort(key=________________)     # BL   Branch_service  (1 point each)
```

(e.2) Fill in the blanks below to reorder `BL` by each Branch's overall satisfaction score, lowest to highest. You may use other functions that have been defined in this exam.

```
________________.sort(key=________________)     # BL   overall_satisfaction  (0.5 for BL, 1.5 for over_sat)
```

(e.3) Write a single Python statement similar to those in (e.1) and (e.2) to reorder BL by each Branch's overall satisfaction score, highest to lowest. You may use other functions that have been defined in this exam.

BL .sort(key=overall_satisfaction, reverse=True)

SCORING: 1.5 pts for reverse-True, 0.5 for everything else correct

Problem 4 (5 points) Topic: Selecting items from a list

In the function definition below, fill in each blank with a single Python constant, operator, or identifier name (variable, function, attribute, method) to satisfy the problem specification.

```
def service_higher (BL: 'list of Branch') -> 'list of Branch':
    """ Return a list of those Branches on the specified list whose service score is
         higher than its drinks score
    """
    result = [ ]
    for b in _____________:                 BL
        if b._____________  _____________ b.drinks:      service   <          b.service > b.drinks
            _____________.append(b)              result
    return _____________                         result
```

SCORING: 1 point per blank.

Problem 5 (6 points) **Topic: for-loop behavior**

Suppose we have a list of Branches called `BL`, as in the previous problem. Fill in each blank with one of the ten code segments below (A through J) to match its most accurate description. Don't use any segment (A–J) more than once.

A.
```
for b in BL:
    print(b.location, b.overall_satisfaction)
```

B.
```
for b in BL:
    if b.ambience >= 75:
        print(b)
```

C.
```
for b in BL:
    if overall_satisfaction(b) > 90:
        print(b.location, b.ID)
```

D.
```
for b in BL:
    print(b.location, overall_satisfaction(b))
```

E.
```
for b in sorted(BL, key=overall_satisfaction, reverse=True):
    print(b.location, b.ID)
```

F.
```
for b in BL:
    if b.ambience >= 75:
        print(b.location)
```

G.
```
for b in BL:
    print(b)
```

H.
```
for b in BL:
    print(BL[b].location, overall_satisfaction(BL[b]))
```

I.
```
for b in BL:
    print(b.location, overall_satisfaction(b) > 90)
```

J.
```
for b in BL:
    print(b.location)
```

SCORING: 1 point each

___ Produces an error message about the improper use of `overall_satisfaction` **(A)**

___ Prints the location and ID of each Branch with overall satisfaction over 90 **(C)**

___ Prints the location of each Branch, one per line **(J)**

___ Prints the location and overall satisfaction score for every Branch **(D) (H has non-num index)**

___ Prints the locations of all Branches whose ambience is at least 75 **(F) (B prints all branch info)**

___ Prints all the information about each Branch in Python namedtuple form **(G)**

Problem 6 (6 points) **Topic: Identifying data types**

Identify the data type of each of the following expressions, using definitions that appear in this exam.

Chose from: `int` `float` `bool` `str` `Branch` list of `Branch` list of `str` list of `float` function

`'party' in L`	**bool**
`overall_satisfaction(B2)`	**float**
`a[0:3] + '!'`	**str**
`L[3:]`	**list of str**
`len(L[4:])`	**int**
`2015`	**int**
`L[1]`	**str**
`B2`	**Branch**
`B2.drinks`	**float**
`[B2, B1]`	**List of Branch**
`a[1] == 'i'`	**bool**
`Branch_service`	**function**

Problem 7 (9 points) **Topic: Control flow and functions**

What does the following code print out?

```
def chocolate (s: str) -> str:
    return "Choc." + s

def icing (n: int, w: str) -> str:
    return chocolate(w) * n

print("Here we go.")
print(chocolate("cake"))
print(icing(3, "blue"))
print("There we are.")
```

```
Here we go.
Choc.cake
Choc.blueChoc.blueChoc.blue
There we are.
```

# SCORING: "Here we go." line first and "There we are." line last: 1pt
"Choc.cake" line correct: 2pts (partial okay)
"Choc.blue" at least once: 2pts
"Choc.blueChoc.blueChoc.blue:: repeated 3 times: 2pt
Everything else correct: 2pts (deduct for extraneous output)
## Check for other versions

<u>When you're done, please</u>:

- Gather up all your stuff.
- Take your stuff and your exam down to the front of the room.
- Turn in your exam; show your ID if asked.
- Exit by the doors at the front of the room. Don't go back or disturb students still taking the test.