ICS 31 • UC IRVINE
WINTER 2014 • DAVID G. KAY

YOUR NAME ______________________________ YOUR LAB:
YOUR STUDENT ID (8 DIGITS) _________________ SECTION (1-9) ______________
YOUR UCINET ID ___________________________ TIME MWF AT: 8 10 12 2 4 6
TA'S NAME ___________________

****
****** K E Y ******
****

First Midterm

You have 75 minutes (until the end of the class period) to complete this exam. There are 60 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 any information or materials with classmates during the exam and 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 (10 points)
Problem 2 (9 points)
Problem 3 (16 points)
Problem 4 (14 points)
Problem 5 (5 points)
Problem 6 (6 points)
Total (60 points)

Problem 1 (10 points) Topic: Simple expressions with numbers, lists, strings

Use the following definitions in this problem:

```
s = 'January'
t = 'February'
n = 28
holidays = ['King', 'Lincoln', 'Washington', 'Ground Hog', "Valentine's"]
```

(a) (2 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("Here we go!")          # Here we go!  [OK on this prob. if quotes]  1/2 pt for this and next line together

print("It'll be fun.")        # It'll be fun.  [Apostrophe required.  Allow enclosing quotes, this prob.]

print(s, n)                   # January 28 [no quotes here]          0.5 pt

print(len(holidays))          # 5                                    0.5pt

print((n * 100) + 3)          # 2803  [check if slightly off]        0.5pt
```

(b) (8 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.

```
print(holidays[2])              # Washington          1 pt

print(holidays[3], 'Day')       # Ground Hog Day      1 pt


print(len(s) > 5)             # True                                       1 pt

print(s[0], t[0])               # J F  [give credit on this problem if no space]       1 pt


print(s[-1] == t[-1])           # True  [Okay for this problem if not capitalized]     1 pt

print('an' in s, 'on' in holidays)    # True False                                     1 pt


print(holidays[-1], 'Day')      # Valentine's Day     1 pt

print(t[0:3], n/2)              # Feb 14  (14.0 is OK) 1 pt
```

Problem 2 (9 points) **Topic: Namedtuples**

(a) (3 points) Anteater Florists has hired you to automate its inventory system. You represent each kind of flower in a namedtuple with attributes for the popular name of the flower, the Latin name of the flower, its color, its price (per stem), and the quantity currently in stock. Which of the following could define that namedtuple? Circle *one or more* of A, B, C, D, E, or F; more than one may be correct.

A. `Flower = namedtuple('Flower', 'popular latin color price instock')` **# THIS ONE**

B. `Flower = namedtuple('Moss Rose', 'Rosa Centifolia', 'red', 3.50)`

C. `Flower = namedtuple('Flower', 'popular, Latin, color, per stem, in stock')`

D. `Flower = namedtuple('Flower', 'pop_name latin_name color price instock')` **# THIS ONE**

E. `Flower = namedtuple('Flower', 'name_popular name_latin cost quantity')`

F. `Flower = Flower('China Rose', 'Rosa Chinensis', 'pink', 2.50)`

(b) (3 points) Which of the following creates a new Flower object, following the definition above, to represent a shipment of 100 purple petunias (Latin name Petunia Grandiflora), to sell at $1.50 a stem? Circle *one or more* of A, B, C, D, or E; more than one may be correct.

A. `new_flower = Flower.petunia('Petunia Grandiflora', 'purple', 1.50, 100)`

B. `new_flower = Flower('petunia, Petunia Grandiflora, purple, 1.50, 100')`

C. `new_flower = Flower('petunia', 'Petunia Grandiflora', 'purple', 1.50, 100)` **# THIS ONE**

D. `new_flower = namedtuple('Flower', 'petunia','Petunia Grandiflora','purple',1.50,100)`

E. `new_flower = Flower(100, 'purple', 'petunia', 'Petunia Grandiflora', 1.50)`

(c) (3 points) When we sell some flowers, we have to update the quantity still in stock. We write the `sell_stems()` function below to update the Flower object's inventory after we sell a specified quantity of that flower.

```
def sell_stems(F: Flower, n: int) -> Flower:
    """ Return the flower with its quantity on hand decreased by the specified number.
        If the full number isn't available, sell what we have, leaving 0 in stock."""
    if n > F.instock:
        remaining = 0
    else:
        remaining = F.instock - n
    return ___________________________________

assert sell_stems(new_flower,15) ==Flower('petunia','Petunia Grandiflora','purple',1.50,85)
assert sell_stems(new_flower,115)==Flower('petunia','Petunia Grandiflora','purple',1.50,0)
```

Which of the following could we insert into the blank space to implement sell-stems correctly? Circle one or more of A, B, C, D, or E; more than one may be correct.

A. `F._replace(instock = remaining)` **# THIS ONE**

B. `F._replace(quantity = remaining)`

C. `Flower(F.popular, F.latin_name, F.color, F.price, remaining)`

D. `Flower(F.popular, F.latin_name, F.color, F.price, n)`

E. `F.quantity = remaining`

Problem 3 (16 points) **Topic: Functions, using namedtuples**

You enjoyed the college application process so much that you've decided to give advice to others selecting a college. You collect college information into a namedtuple defined as follows:

```
College = namedtuple('College', 'name location students faculty tuition')
```

The name and location fields are strings; students and faculty are integers representing the number of people in each category; tuition is a float.

(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 student_faculty_ratio(c: College) -> float:
    """ Return the number of students per faculty member at this college
    """
    return _______________ . _______________ / _______________ . _______________
```

return c.students / c.faculty

(b) (3 points) Fill in each blank with a single Python constant, operator, or identifier name so that each assertion will be true according to the data provided (assuming a correct solution to part (a)).

```
UCI = College('UC Irvine', 'Irvine, CA', 22216, 2222, 13122)
Occidental = College('Occidental College', 'Los Angeles, CA', 1607, 211, 41438)

assert ____________ ( _____________ ) == 22216/2222   # 9.998
assert ____________ ( _____________ ) == 1607/211     # 7.616
```

student_faculty_ratio(UCI)

student_faculty_ratio(Occidental)

(c) (4 points) What do the following statements print? (Note that you don't have to do the heavy arithmetic; the values are provided for you above.)

```
def College_to_str(c: College) -> str:
    """ Arrange College information in a readable form, e.g., for printing
    """
    return (c.name + ' in ' + c.location + " costs $" + str(c.tuition) +
        " (S:F ratio " + str(student_faculty_ratio(c)) + ":1)")
print(College_to_str(UCI))
print(College_to_str(Occidental))
```

UC Irvine in Irvine, CA costs $13122 (S:F ratio 9.998:1)
Occidental College in Los Angeles, CA costs $41438 (S:F ratio 7.616:1)

(d) (2 points) Why do we have two calls to the `str()` function in the `return` statement above? Circle the single best, most correct answer from A, B, C, or D.

A. We're using + to concatenate strings; it requires that both its operands be strings. **THIS ONE**

B. We can't create a string that contains numbers.

C. Return statements require us to provide strings.

D. The point of this function is to create one large, printable string representing a college.

(e) (3 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 print_colleges(L: 'list of College') -> None:
    ''' For each college on the list, print its string representation.
    '''
    for c in ______________:     L

        print(______________(______________))     College_to_str     c
```

SCORING: 1 point for each blank

Problem 4 (14 points) **Topic: Processing and sorting lists of namedtuples**

Suppose we have a list of College objects called `CL`. Also recall the definition of `student_faculty_ratio()` in Problem 3. Write one Python statement using `sort()` or `sorted()` to accomplish each of the tasks below. We provide this excerpt from `help()` for reference:

```
sort(...)
       L.sort(key=None, reverse=False) -> None -- stable sort *IN PLACE*
sorted(...)
       sorted(L, key=None, reverse=False) --> new sorted list
```

(a) (3 points) Rearrange `CL` into alphabetical order by college name

CL.sort()

(b) (3 points) Rearrange `CL` into order by student-faculty ratio, lowest to highest

CL.sort(key=student_faculty_ratio)

(c) (3 points) Assign to `CL2` a list of colleges in `CL`, in alphabetical order by college name, without changing `CL`

CL2 = sorted(CL)

(d) (5 points) Assign to `CL2` a list of the colleges in `CL` ordered by student-faculty ratio, highest to lowest, without changing `CL`

CL2 = sorted(CL, key=student_faculty_ratio, reverse=True)

Problem 5 (5 points) **Topic: Identifying types**

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

Choose from: `int` `float` `bool` `str` `College` list of `College` list of `str` list of `float` function

`(n * 100) + 3`	**int (or float)**	`Occidental.tuition < UCI.tuition`	**bool**
`new_flower.color`	**str**	`holidays`	**list of str**
`len(holidays)`	**int**	`holidays[-1]`	**str**
`Occidental`	**College**	`student_faculty_ratio`	**function**
`Occidental.tuition`	**float**	`sorted(CL)`	**list of College**

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

Suppose we have a list of Colleges called `CL`, as in a previous problem. Match the six for-loops below (A through F) with the most accurate description below.

A.
```
for c in CL:
    print(c)
```

B.
```
for c in sorted(CL, key=student_faculty_ratio):
    print(c.name, c.location)
```

C.
```
for c in CL:
    if c.tuition < 10000:
        print(c.name, c.location)
```

D.
```
for c in sorted(CL, key=student_faculty_ratio):
    print(c.name, c.student_faculty_ratio)
```

E.
```
for c in CL:
    print(c.name, c.location, c.tuition * c.students)
```

F.
```
for c in CL:
    print(College_to_str(c))
```

___ Produces an error message **(D)**

___ Prints all the stored attributes (fields) of each College in Python namedtuple form **(A)**

___ Prints some information about each college, in order by student-faculty ratio **(B)**

___ Prints each college's information in a form that's easy for users to read **(F)**

___ Produces the total tuition each college collects, assuming every student pays full tuition **(E)**

___ Prints the name and location of colleges that charge less than $10,000 tuition **(C)**

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

- 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.