ICS 31 • UC Irvine
Spring 2017 • David G. Kay

Your Name _____

Your Student ID (8 digits) _____

Your UCInet ID _____@uci.edu

##############
KEY
##############

First Miderm

Your Lab Section (circle one):

1. 8:00 Ashwin Achar
2. 9:30 Ashwin Achar
3. 11:00 Nathaniel Baer
4. 12:30 Nathaniel Baer
5. 2:00 Yadhu Prakash
6. 3:30 Yadhu Prakash
7. 5:00 Shreyaan Kaushal
8. 6:30 Shreyaan Kaushal

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 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 (10 points)	
Problem 2 (9 points)	
Problem 3 (16 points)	
Problem 4 (6 points)	
Problem 5 (6 points)	
Problem 6 (13 points)	
Total (60 points)	

Problem 1 (10 points) **Topic: Evaluating expressions**

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

```
THIS_YEAR = 2017                    # We gave credit in some places for variations in quotes, spacing,
s = "spring"                        # or capitalization, but that doesn't mean they're strictly right.
f = ['banana', 'plum', 'strawberry', 'rhubarb', 'apricot', 'pineapple']
```

(a) (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.)

```
print("It's time.")          # It's time. [OK with or without quotes, but apostrophe required]
print("Go! Go! Go!")        # Go! Go! Go!  [Quotes surrounding OK]   0.5 pts for both this line and prior
print("Spring", THIS_YEAR)  # Spring 2017 [No credit for "Spring" 2017; OK if no space before 2017] 0.5 pts
print(200 + THIS_YEAR)      # 2217        0.5 pts
print(len(f) / 2)           # 3.0  or  3  0.5 pts
print(len(s) * 3)           # 18          0.5 pts
print(THIS_YEAR <= 2000)    # False       0.5 pts
```

(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.)

```
print(s[0])                 # s (with quotes OK)      0.5
print(s[-1] == 'n')         # False                   0.5
print(s[2] + 'ed')          # red (with quotes OK)    0.5
print('ring' in s)          # True                    0.5
print('f' in 'alpha')       # False                   0.5
print('UCI'*3, len('UCI'*3))# UCIUCIUCI  9    0.5  One set of quotes around UCIUCIUCI OK, 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.)

```
print(f[0], f[-1])          # banana pineapple        1 point  Quotes around each, OK; otherwise no.
print(len(f), len(f[1:]))   # 6  5                    1 point
print(f[1], f[3], f[5])     # plum rhubarb pineapple  1 point.  Quotes around individual words OK.
print('Apricot' in f)       # False                   1 point  Upper case and lower case are different.
```

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

(a) (3 points) The Antflix online video service represents each movie it rents in a namedtuple called Film that has four fields: the title of the movie, its length in minutes, the year it was made, and the price for a 24-hour rental. Which of the following defines a namedtuple that satisfies this specification? Circle *one or more* of A, B, C, D, or E; more than one may be correct.

A. `Film = namedtuple('Film', 'title length year rental')` **# This one**

B. `Film = namedtuple('Film', 'title of film, minutes long, year made, rental price')`

C. `Film = namedtuple('Film', 'Star Wars', 121, 1977, 3.50)`

D. `Film = namedtuple('Film', 'name length year_made rate')` **# This one**

E. `Film = Film('Star Wars 121 1977 3.50')` **SCORING: See below.**

(b) (3 points) Which of the following creates a Film object as a namedtuple following the description above, to represent the 91-minute film "The Computer Wore Tennis Shoes," made in 1969 and renting for $2.00? Circle *one or more* of A, B, C, D, or E; more than one may be correct.

A. `film1 = Film('The Computer Wore Tennis Shoes', 91, 1969, 2.00)` **# This one**

B. `film1 = The_Computer_Wore_Tennis_Shoes(91, 1969, 2.00)`

C. `film1 = Film.The_Computer_Wore_Tennis_Shoes(91, 1969, 2.00)`

D. `film1 = Film.length * Movie.price`

E. `film1 = Film(91, 'The Computer Wore Tennis Shoes', 1969, 2.00)`

SCORING: –1 for each mistake (wrong answer circled or right answer uncircled); no negative scores.

(c) (3 points) You want to insert a two-minute advertisement at the beginning of each movie. Which of the following statements correctly increases the length of the Film object stored in `a_film`? [You may assume any of the correct Film definitions above.] Circle *one or more* of A, B, C, D, or E; more than one may be correct.

A. `a_film.length = a_film.length + 2`

B. `a_film = a_film._replace(length = a_film.length + 2)` **# This one**

C. `a_film = a_film.replace(length = 92)`

D. `a_film = Film(length = a_film.length + 2)`

E. `a_film = Film(a_film.title, a_film.length + 2, a_film.year, a_film.rental)` **# This one**

Problem 3 (16 points) **Topic: functions, sorting a list of namedtuples**

Antflix has hired you to implement star-rating scores for each movie. Antflix members give each movie a 0-to-4 score on each of five categories: acting, directing, writing, costumes, and music. Antflix uses the following namedtuple to represent Movie objects:

```
Movie = namedtuple('Movie', 'title year acting directing writing costumes music')
```

The title is a string, the year is an int, and the remaining fields are floats, storing the average rating from all Antflix members on each category for that movie.

(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 star_rating(m: Movie) -> float:
  ''' Return the movie's overall score, in the range 0-4, computed as an equally-
      weighted average of its acting, directing, writing, costumes, & music scores.
  '''
  return (m.acting + m._____ + _____.writing _____ m.costumes + m.music) _____ 5
```

return (m.acting + m.directing + m.writing + m.costumes + m.music) / 5 # directing m + / 1pt each

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

```
assert star_rating(Movie('Brazil', 1985, 3, 3, 3, 3, 3)) == _____
assert star_rating(Movie('Brazil', 1985, _____, _____, _____, _____, _____)) == 0
```

First line: 3 (0.5 pt) Second line: 0 0 0 0 0 (0.5 pt)

(c) (5 points) What do the following statements print? [You can do the arithmetic in your head.]

```
m1 = Movie('The Meaning of Life', 1983, 4, 4, 4, 4)
m2 = Movie('Life of Brian', 1979, 4, 4, 2, 2, 3)
print(m1.title, "(", m1.year, ") Rating", star_rating(m1))
print(m2.title, "(", m2.year, ") Rating", star_rating(m2))
```

The Meaning of Life (1983) Rating 4
Life of Brian (1979) Rating 3
SCORING: 1/2 point for each name, 1/2 pt for each year,
1 point for each correct number with the right name (omitting or adding .0 okay),
1 point for everything else correct (without worrying about extra spacing)

(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 Movie_acting(m: Movie) -> float:
    ''' Return a movie's score for acting
    '''
    return _____ . _____        # m . acting
```

(e) (4 points) In addition to the function `Movie_acting` defined above, assume you have also defined similar functions called `Movie_directing`, `Movie_writing`, `Movie_costumes`, and `Movie_music` that return the values of those fields. Also, suppose that Antflix has a list of 2500 movies called `ML`.

(e.1) Fill in the blanks below to reorder `ML` by each Movie's directing score, lowest to highest.

```
_____.sort(key= _____)      # ML  Movie_directing  (1 point each)
```

(e.2) Fill in the blanks below to reorder `ML` by each Movie's overall star-rating. You may use other functions that have been defined in this exam.

```
_____.sort(key= _____)      # ML  star_rating (0.5 for ML, 1.5 for star_rating)
```

Problem 4 (6 points) Topic: Identifying types

Identify the data type (not the value) of each of the following expressions, using definitions that appear in this exam.

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

`s[1] + 'ie'`	**str**
`len(f[1:])`	**int**
`1492`	**int**
`f[1]`	**str**
`m2.music`	**float**
`m1`	**Movie**
`s[1] == 'e'`	**bool**
`Movie_acting`	**function**
`ML`	**list of Movie**
`'gooseberry' in f`	**bool**
`star_rating(m2)`	**float**
`f[1:]`	**list of str**

Problem 5 (6 points) **Topic: for-loops**

Suppose we have a list of Movies called `ML`, as in the previous problem. Match the six for-loops below (A through F) with the most accurate description below. (Use each loop and each description exactly once.)

A.
```
for m in ML:
    print(m.title)
```

B.
```
for m in ML:
    print(m.title, star_rating(m))
```

C.
```
for m in ML:
    if m.year < 2000:
        print(m.title)
```

D.
```
for m in ML:
    print(m, star_rating(m))
```

E.
```
for m in ML:
    if star_rating(m) >= 3:
        print(m.title, m.year)
```

F.
```
for m in ML:
    print(m.title, m.star_rating)
```

___ Print the name of each Movie with its overall star rating **...(m.title, star_rating(m))...**

___ Produces an error message about the improper use of `star_rating` **...(m.title, m.star_rating...**

___ Produces everything stored about each Movie **... (m, star_rating(m))...**

___ Prints the names of the movies made before 2000. **... if m.year < 2000 ...**

___ Print the names of the Movies, one per line **...(m.title)...**

___ Prints the names and years of the Movies with three or more stars. **... if star_rating(m) >= 3 ...**

B F D C A E

Problem 6 (13 points) **Topic: Control flow**

(a) (4 points) What does the following code print out?

```
print('Spring fruits')
for fruit in f:  # Use the definition of f from Problem 1
    if fruit[0] > 'q':
        print(fruit)
print('Farmers markets')
```

Spring fruits
strawberry
rhubarb
Farmers markets

SCORING:
1/2 pt for Spring fruits once at top (don't worry about capitalization)
2 pts for correct names in correct order
1/2 pt for Farmers markets once at bottom
up to 1 pt for everything else correct

(b) (4 points) What does the following code print out?

```
print("April Fools Day")
year = 2016
for n in range(6):
    print(year, year + n)
print('May Day')
```

April Fools Day
2016 2016
2016 2017
2016 2018
2016 2019
2016 2020
2016 2021
May Day

SCORING:
1/2 point for April Fools Day at top
1/2 point for May Day at bottom
1/2 point for six lines of two numbers each
1/2 point for first column being all 2016s
1/2 point for 2d col. being numbers > 2010 and < 2100, increasing by 1 each line
1/2 point for second column starting at 2016 (not 2017)
up to 1 pt for everything else correct

(c) (5 points) What does the following code print out? [Reminder: In Python, multiplying a string by a number produces a new string containing that many copies of the first string, so `3 * "OK"` is `"OKOKOK"`.]

```
def n_copies (n: int, s: str) -> str:
    return  s * n

print("Memorial Day")
spring = n_copies(2, 'peace')
print(spring, "//", n_copies(2, spring))
print("Flag Day")
```

Memorial Day
peace peace // peace peace peace peace
Flag Day

SCORING:
1/2 pt for Memorial Day at top
1/2 pt for Flag Day at bottom
1 pt for at least one 'peace', //, then at least one 'peace' [no quotes]
1 pt for two copies (OK if spaces between words, as shown above, though no-spaces is correct) before //
1 pt for four copies (same policy for spacing between words) after //
1 pt for everything else correct

Problem 7 (0 points)

When you're done with the exam, follow these steps (so you don't disturb your classmates and so your exam gets turned in properly):

- Write your UCInet ID in the blanks at the top of the odd-numbered pages. Also check for your name on the front page.
- 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 to your seat or disturb students who are still working.