ICS 31 • UC IRVINE
FALL 2013 • DAVID G. KAY

****

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

****

YOUR NAME ______________________________ YOUR LAB:

YOUR STUDENT ID (8 DIGITS) ________________ SECTION (1-14) ______________

YOUR UCINET ID __________________________ TIME MWF AT: 8 10 11 12 2 4 6

TA'S NAME __________________

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, including cellphones.

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)

37aMwx3

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

Use the following definitions in this problem:

```
THIS_YEAR = 2013
p = "autumn"
R = ['carrot', 'turnip', 'beet', 'rutabaga', 'radish', 'parsnip', 'daikon']
```

(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("Ready?")              # Ready? [OK with or without quotes]     0.5 pts for both this line and the next
print("Don't panic!")        # Don't panic!  [Quotes surrounding OK, but apostrophe required]
print("Fall", THIS_YEAR)     # Fall 2013 [No credit for "Fall" 2013; OK if no space before 2013] 0.5 pts
print(100 + THIS_YEAR)       # 2113    0.5 pts
print(10 * len(p))           # 60      0.5 pts
print(len(p) / 2)            # 3       0.5 pts
print(THIS_YEAR >= 2000)     # True    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(p[0])                  # a (with quotes OK)     0.5
print(p[-1] == 'e')          # False                  0.5
print(p[2] + 'ea')           # tea (with quotes OK)   0.5
print('tea' in p)            # False                  0.5
print('x' in 'uvwxyz')       # True                   0.5
print('ICS'*2, len('ICS'*2)) # ICSICS  6       0.5  One set of quotes around ICSICS 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(R[0], R[-1])           # carrot daikon               1 point  Quotes around each, OK; otherwise no.
print(len(R), len(R[1:]))    # 7 6                         1 point
print(R[1], R[3], R[5])      # turnip rutabaga parsnip     1 point.  Quotes around individual words OK.
print('DAIKON' in R)         # False                       1 point   Upper case and lower case are different.
```

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

(a) (3 points) The Zotflix online video service represents each movie it rents in a namedtuple called Film that has four fields: the title of the movie, the year it was made, its length in minutes, 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 year length rental')` **# This one**

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

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

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

E. `Film = Film('Star Wars 1977 121 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', 1969, 91, 2.00)` **# This one**

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

C. `film1 = Film.The_Computer_Wore_Tennis_Shoes(1969, 91, 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 three-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 = a_film._replace(length = a_film.length + 3)` **# This one**

B. `a_film.length = a_film.length _ 3`

C. `a_film = a_film.replace(length = 94)`

D. `a_film = Film(a_film.title, a_film.year, a_film.length + 3, a_film.rental)` **# This one**

E. `a_film = Film(length = a_film.length + 3)`

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

Zotflix has hired you to implement star-rating scores for each movie. Zotflix members give each movie a 0-to-4 score on each of five categories: acting, directing, writing, costumes, and music. Zotflix 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 Zotflix 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, 4, 4, 4, 4, 4)) == _______
assert star_rating(Movie('Brazil', 1985, _______, _______, _______, _______, _______)) == 0
```

First line: 4 (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, 3, 3, 3, 3, 3)
m2 = Movie('Life of Brian', 1979, 4, 4, 3, 3, 3.5)
print(m1.title, "(", m1.year, ") Rating", star_rating(m1))
print(m2.title, "(", m2.year, ") Rating", star_rating(m2))
```

The Meaning of Life (1983) Rating 3
Life of Brian (1979) Rating 3.5
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 Zotflix has a list of 2500 movies called `ML`.

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

```
_______________.sort(key= _______________)      # ML  Movie_writing  (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 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

`'DAIKON' in R`	**bool**
`star_rating(m2)`	**float**
`p[1] + 'es'`	**str**
`len(R[1:])`	**int**
`1492`	**int**
`R[1]`	**str**
`m2.music`	**float**
`m1`	**Movie**
`p[1] == 'e'`	**bool**
`Movie_acting`	**function**
`ML`	**list of Movie**
`R[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.

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)
```

___ 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 since 2000. **... if m.year > 2000 ...**

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

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

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

F D C A B E

Problem 6 (13 points) Topic: Control flow

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

```
print('Root vegetables')
for v in R:   # Use the definition of R from Problem 1
    if v[0] < 'e':
        print(v)
print('Winter produce')
```

Root vegetables	**SCORING:**
carrots	**1/2 pt for Root vegetables once at top (don't worry about capitalization)**
beets	**2 pts for correct names in correct order**
daikon	**1/2 pt for Winter produce once at bottom**
Winter produce	**up to 1 pt for everything else correct**

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

```
print("Columbus Day")
year = 1492
for n in range(5):
    print(year, year + n)
print('Discover America')
```

Columbus Day
1492 1492
1492 1493
1492 1494
1492 1495
1492 1496
Discover America

SCORING:
1/2 point for Columbus Day at top
1/2 point for Discover America at bottom
1/2 point for five lines of two numbers each
1/2 point for first column being all 1492s
1/2 point for 2d col. being numbers > 1490 and < 1500, increasing by 1 each line
1/2 point for second column starting at 1492 (not 1493)
up to 1 pt for everything else correct

(c) (5 points) What does the following code print out?

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

print("Halloween")
scary = n_copies(2, 'ghost')
print(scary, "***", n_copies(2, scary))
print("Boo!")
```

Halloween
ghost ghost * ghost ghost ghost ghost**
Boo!

SCORING:
1/2 pt for Halloween at top
1/2 pt for Boo! at bottom
Version 1 pt for at least one 'ghost', *, then at least one 'ghost' [Quotes shouldn't be there)**
other versions: ghost goblin witch monster skeleton zombie
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