ICS 31 • UC Irvine
Winter 2015 • David G. Kay
********** KEY **********

Your Name _____
Your Student ID (8 digits) _____
Your UCInet ID _____

Your Lab Section (circle one):

1. 8:00 Sanket Khanwalkar
2. 10:00 Neeraj Kumar
3. 12:00 Sanket Khanwalkar
4. 2:00 Vignesh Raghunathan
5. 4:00 Vignesh Raghunathan
6. 6:00 Neeraj Kumar
7. 10:00 Andrea D'Souza
8. 2:00 Andrea D'Souza

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 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	(5 points)
Problem 5	(6 points)
Problem 6	(6 points)
Problem 7	(8 points)
Total	(60 points)

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

Use the following definitions in this problem:

```
THIS_YEAR = 2015
p = "winter"
L = ['sweaters', 'skis', 'hats', 'ice skates', 'hot tea', 'snow']
```

(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 to go!")      # Ready to go!
print("It's not cold.")    # It's not cold. [Apostrophe required.]
print(p, THIS_YEAR)        # winter 2015 [no quotes here]
print(100 + THIS_YEAR)     # 2115 [no quotes here]
print(len(L) / 2)          # 3.0 or 3
print(THIS_YEAR >= 2000)   # True
```

(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(p[0])                # w
print(p[-1] == 'e')        # False
print(p[1] + 'nk')         # ink
print('int' in p)          # True
print(p in 'seasons')      # False
print('ICS'*2, len('ICS'*2))   # ICSICS  6
```

(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[0], L[-1])                 # sweaters  snow
print(len(L), len(L[1:]))          # 6  5
print(L[1], L[3], L[5])            # skis  ice skates  snow
print("HOT TEA" in L)              # False
```

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

The ICStunes Music Store represents each song in namedtuple called Song that has four fields: the song title, the artist's name, the year it was produced, 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. `Song = namedtuple('Song', 'title artist year price')` **# THIS ONE**

B. `Song = namedtuple('Song', 'Beat It', 'Michael Jackson', 0.99, 1982)`

C. `Song = namedtuple('Song', 'song title, artist name, year made, price')`

D. `Song = namedtuple('Song', 'title artist_name year_produced download_price')` **# THIS**

E. `Song = namedtuple('Song', 'title artist price')`

F. `Song = Song('Thriller', 'Michael Jackson', 1983, 1.29)`

(b) (3 points) Which of the following creates a new Song object, following the definition above, to represent Michael Jackson's song "Bad" that came out in 1987 and sells for $0.99? Circle *one or more* of A, B, C, D, or E; more than one may be correct.

A. `new_song = Song.Bad('Michael Jackson', 1987, 0.99)`
B. `new_song = Song('Bad, Michael Jackson, 1987, 0.99')`
C. `new_song = Song('Bad', 'Michael Jackson', 1987, 0.99)` **# THIS ONE**
D. `new_song = namedtuple('Song', 'Bad', 'Michael Jackson', 1987, 0.99)`
E. `new_song = Song(0.99, 'Bad', 'Michael Jackson', 1987)`

(c) (3 points) You want to increase the price of each song by ten cents. Which of the following statements correctly increases the price of the Song object stored in `a_song` (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_song = a_song._replace(price = a_song.price + 0.10)` **# THIS ONE**
B. `a_song.price = a_song.price + 0.10`
C. `a_song = Song(a_song.title, a_song.artist, a_song.year, a_song.price + 0.10)` **#THIS ONE**
D. `a_song = a_song.replace(price = 1.09)`
E. `a_song = Song(price = a_song.price + 10)`
F. `print(a_song.price + 0.10)`

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

ICStunes has hired you to implement star-rating scores for each song. ICStunes members give each movie a 0-to-5 score on each of three categories: vocals, music, and danceability. ICStunes uses this namedtuple to represent rated songs:

```
RatedSong = namedtuple('RatedSong', 'title year vocals music danceability')
```

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

(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(rs: RatedSong) -> float:
    """ Return the song's overall star-rating, in the range 0-5, computed as an
        equally-weighted average of its vocals, music, and danceability scores.
    """
    return (rs.vocals __________ rs.__________ + __________.danceability) __________ 3
```

return (rs.vocals + rs.music + rs.danceability) / 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 star_rating(RatedSong('All About That Bass', 2014, 3, 3, 3)) == _____

assert star_rating(RatedSong('All About That Bass', 2014, ____, _____, ____)) == 0
```

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

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

```
S1 = RatedSong('Sugar Sugar', 1969, 4, 4)
S2 = RatedSong('Born to be Wild', 1968, 2, 3, 2.5)
print(S1.title, "(", S1.year, ") Rating", star_rating(S1))
print(S2.title, "(", S2.year, ") Rating", star_rating(S2))
```

Sugar Sugar (1969) Rating 4
Born to be Wild (1968) Rating 2.5

(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 Song_vocals(rs: RatedSong) -> float:
    ''' Return a song's score for vocals
    '''
    return __________ . __________
```

 # rs vocals

(e) (4 points) In addition to the function `Song_vocals` defined above, assume you have also defined similar functions called `Song_music` and `Song_danceability` that return the values of those fields. Also, suppose that ICStunes has a list of 25,000 songs called `SL`.

(e.1) Fill in the blanks below to reorder `SL` by each song's music score, lowest to highest:

```
_________________.sort(key=_________________)
```

 # SL Song_music (1 point each)

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

```
_________________.sort(key=_________________)
```

 # SL star_rating (0.5 for SL, 1.5 for star_rating)

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 top_rated_songs (SL: 'list of RatedSong') -> 'list of RatedSong':
    """ Return a list of those songs on the input list whose overall star rating
        is 4 or greater.
    """
    result = [ ]
    for s in _____________:                SL
        if _____________(s) _____________ 4:     star_rating    >=
            _____________.append(s)              result
    return _____________                         result
```

SCORING: 1 point per blank.

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

Suppose we have a list of RatedSongs called `SL`, 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 s in SL:
    print(s.title)
```

B.
```
for s in SL:
    print(s.title, s.star_rating)
```

C.
```
for s in SL:
    if s.year > 2000:
        print(s)
```

D.
```
for s in SL:
    if star_rating(s) >= 3:
        print(s.title, s.year)
```

E.
```
for s in SL:
    print(s.title, star_rating(s))
```

F.
```
for s in sorted(SL, key=star_rating, reverse=True):
    print(s.title, s.year)
```

G.
```
for s in SL:
    if s.year > 2000:
        print(s.title)
```

H.
```
for s in SL:
    print(s)
```

I.
```
for s in SL:
    print(SL[s].title, star_rating(SL[s]))
```

J.
```
for s in SL:
    print(s.title, star_rating(s) >= 3)
```

SCORING: 1 point each

___ Produces an error message about the improper use of `star_rating` **(B)**

___ Prints the title and year of each song with three or more stars **(D)**

___ Prints the title of each song, one per line **(A)**

___ Prints the title and number of stars for every song **(E) (i has non-num index)**

___ Prints the titles of all songs recorded after 2000 **(G) (c prints all song info)**

___ Prints all the information about each song in Python namedtuple form **(H)**

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` `RatedSong` list of `RatedSong` list of `str` list of `float` function

`'skates' in L`	**bool**
`star_rating(S2)`	**float**
`p[0:3] + 'e'`	**str**
`len(L[3:])`	**int**
`2015`	**int**
`L[1]`	**str**
`S2.vocals`	**float**
`S2`	**RatedSong**
`[S2, S1]`	**List of RatedSong**
`p[1] == 'i'`	**bool**
`Song_vocals`	**function**
`L[3:]`	**list of str**

Problem 7 (8 points) **Topic: Control flow and functions**

What does the following code print out?

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

print("Last problem!")
first = n_copies(2, 'cat')
print(first, '***', n_copies(2, first))
print("The end.")
```

Last problem!
cat cat * cat cat cat cat** **[Note versions with different animals, different separators]**
The end.
SCORING: 1 point for first line. 1 point for last line.
2 points for at least one 'cat', '*', then at least one 'cat' [with no apostrophes; they're just here for clarity]**
1 point for two copies before * [spaces between words doesn't lose points]**
2 points for four copies after * [same policy for spacing between words]**
1 point for everything else correct

When you're done, please:

- 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.