ICS 31 • A • UC Irvine
Winter 2016 • David G. Kay
****
****** KEY ******
****

Your Name ______________________
Your Student ID (8 digits) ______________
Your UCInet ID ______________________

Your Lab Section (circle one):

1.	8:00a	Shibani Konchady
2.	10:00	Shibani Konchady
3.	4:00	Akshat Amrish Patel
4.	6:00	Akshat Amrish Patel
5.	8:00a	Aniket Shivam
6.	10:00	Aniket Shivam
7.	4:00	Roeland Singer-Heinze
8.	6:00	Roeland Singer-Heinze

Second Midterm

You have 75 minutes (until the end of the class period) to complete this exam. There are 55 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 (5 points)	
Problem 2 (21 points)	
Problem 3 (9 points)	
Problem 4 (15 points)	
Problem 5 (5 points)	
Total (55 points)	

Problem 1 (5 points) **Topic: Data Types**

A library represents each book in its collection as follows:

```
from collections import namedtuple
Book = namedtuple('Book', 'callnum author title year pages checkedout')
bk6 = Book("y153.w", "Sayers, Dorothy", "Whose Body?", 1923, 156, False)
bk7 = Book("z13.21", "Sayers, Dorothy", "Busman’s Honeymoon", 1937, 381, True)
bk8 = Book("w22.45a", "Doyle, Arthur Conan", "His Last Bow", 1917, 225, False)
bk9 = Book("x12.5a", "Doyle, Arthur Conan", "A Study in Scarlet", 1887, 325, True)
BL = [bk6, bk7, bk8, bk9]
```

The field `callnum` is a string with the book's call number (a unique ID for each book). The author and title are strings; the year of publication and the number of pages are ints; the `checkedout` field is boolean: True if the book is checked out and False if it's still at the library.

The Orange County Public Library system represents each of its branch libraries as follows:

```
Library = namedtuple('Library', 'name address phone collection')
ElTo = Library("El Toro", "24672 Raymond Way, El Toro, CA  92630", "949-855-8173",
                 [bk6, bk7, bk8])
Brea = Library("Brea", "1 Civic Center Circle, Brea, CA  92821", "714-671-1722",
             [bk7, bk8, bk9])
IrvUP = Library("Irvine University Park", "4152 Sandburg Way, Irvine, CA  92612",
                 "949-786-4001", [bk6])
Tust = Library("Tustin", "345 East Main Street, Tustin, CA  92780", "714-544-7725",
                 [bk6, bk9])
OCPL = [ElTo, Brea, IrvUP, Tust]
```

These definitions will be used throughout this test.

For each of these expressions, (i) check the box corresponding to its data type and (ii) if it's a list, give its length; if it's a namedtuple, give the value of its first field; otherwise, give its value.

(a) ❑int ❑float ❑bool ❑str ❑list of str ❑Book ❑Library ❑list of Book ❑list of Library

`BL[1].year` **int, 1937**

(b) ❑int ❑float ❑bool ❑str ❑list of str ❑Book ❑Library ❑list of Book ❑list of Library

`OCPL` **list of Library, 4**

(c) ❑int ❑float ❑bool ❑str ❑list of str ❑Book ❑Library ❑list of Book ❑list of Library

`Brea.collection` **list of Book, 3**

(d) ❑int ❑float ❑bool ❑str ❑list of str ❑Book ❑Library ❑list of Book ❑list of Library

`ElTo.collection[2].pages` **int, 225**

(e) ❑int ❑float ❑bool ❑str ❑list of str ❑Book ❑Library ❑list of Book ❑list of Library

`OCPL[0].collection[1]` **Book, z13.21**

Problem 2 (21 points) **Topic: Processing lists of namedtuples.**

(Continue using the definitions from Problem 1.)

(a) (4 points) Complete the definition of the function below, consistent with its header, docstring comment, and assertions, by filling in each blank with exactly one identifier, operator, or constant.

```
def book_from_callnum(booklist: 'list of Book', call_number: str) -> Book:
    """ Return the Book with the specified call number, or None if not found.
    """
    for bk in ______________________:  booklist
        if bk.______________________ == ______________________:  callnum == call_number
            return ______________________  bk  [versions: bk, a_book, this_bk, the_book]
    return None

assert book_from_callnum(BL, "z13.21") == bk7
assert book_from_callnum(BL, "xyz123") == None
```

(b) (3 points) Complete the definition of the function below, consistent with its header, docstring comment, and assertions, by filling in each blank with exactly one identifier, operator, or constant.

```
def title_from_callnum(booklist: 'list of Book', call_number: str) -> str:
    """ Given a list of Books and a call number, return the title of the book with
        that call number
    """
    bk = ______________________ (booklist, call_number)   book_from_callnum  (2 pts)
    return bk.______________________  title

assert title_from_callnum(BL, "z13.21") == "Busman's Honeymoon"
```

(c) (4 points) Complete the definition of the function below, consistent with its header, docstring comment, and assertions, by filling in each blank with exactly one identifier, operator, or constant.

```
def books_checked_out(booklist: 'list of Book') -> 'list of Book':
    """ Return a list of those books in the parameter that are checked out
    """
    result = [ ]
    for bk in booklist:
        if bk.______________________:  checkedout
            ______________________.append(______________________)  result.append(bk)
    return ______________________  result

assert books_checked_out(BL) == [bk7, bk9]
assert books_checked_out([ ]) == [ ]
```

(d) (4 points) Complete the definition of the function below, consistent with its header, docstring comment, and assertions, by filling in each blank with exactly one identifier, operator, or constant.

```
def percentage_checked_out(booklist: 'list of Book') -> float:
    """ Out of the total number of books on the list, the percentage (0-100)
        that are currently checked out. """
    return len(______________________(______________________)) /   \

           ______________________ (______________________) * 100

assert percentage_checked_out(BL) == 50   len(books_checked_out(booklist)) / len(booklist) * 100
assert percentage_checked_out([b7, b9]) == 100
assert percentage_checked_out([b6, b8]) == 0
```

(e) (6 points) Complete the definition of the function below, consistent with its header, docstring comment, and assertions, by filling in each blank with exactly one identifier, operator, or constant.

```
def book_length(bk: ____________________) -> int:    Book
    """ Return the number of pages in the book
    """
    return bk.____________________ pages

def longest_book_available(booklist: 'list of Book') -> Book:
    """ Return the longest Book that is currently not checked out.
    """
    result = [ ]
    for bk in booklist:
        if not bk.____________________: checkedout
            result.append(____________________) bk  [check versions]
    result.sort(key = ____________________, reverse = True) book_length
    return result[____________________] 0
assert longest_book_available(BL) == bk8
```

Problem 3 (9 points) Topic: String processing

(Continue using the definitions in Problem 1.) The following excerpt from `help(str)` may be useful for this problem.

```
find(...)
    S.find(sub) -> int
    Return the lowest index in S where
    substring sub is found.
    Return -1 on failure.

join(...)
    S.join(list) -> str
    Return a string which is the
     concatenation of the strings in the
    list.  The separator between elements
     is S.

replace(...)
    S.replace(old, new) -> str
    Return a copy of S with all
     occurrences of substring old
     replaced by new.

split(...)
    S.split([str]) -> list of strings
    Return a list of the words in S,
     using str as the delimiter string.
     If str is not specified or is None,
     any whitespace string is a separator
     and empty strings are removed from
     the result.

strip(...)
    S.strip([chars]) -> str
    Return a copy of the string S with
     leading and trailing whitespace
     removed.  If chars is given and not
     None, remove characters in chars
     instead.
```

The address field of a Library contains the whole mailing address in one string. The code below constructs a namedtuple with a separate field for each component of the address.

```
Address = namedtuple('Address', 'street_addr city state zip')

def string_to_Address(entire_addr: str) -> Address:
    """ Create an Address with the contents of the string
    """
    three_parts = entire_addr.split(",")
    street_addr = three_parts[0].strip()
    city = three_parts[1].strip()
    state_zip = three_parts[2].split()
    state = state_zip[0]
    zip = state_zip[1]
    return Address(street_addr, city, state, zip)

assert string_to_Address("5056 Donald Bren Hall, Irvine, CA  92697") == \
     Address("5056 Donald Bren Hall", "Irvine", "CA", "92697")
assert string_to_Address("   24672 Raymond Way,    El Toro,CA 92630   ") == \
     Address("24672 Raymond Way", "El Toro", "CA", "92630")
```

(a) (2 points) When we call `string_to_Address("5056 Donald Bren Hall, Irvine, CA  92697")`, what is the value assigned to `three_parts`?

A. `["5056", "Donald", "Bren", "Hall", "Irvine", "CA", "92697"]`

B. `["5056 Donald Bren Hall, Irvine, CA  92697"]`

C. `["5056 Donald Bren Hall", " Irvine", " CA  92697"]` **# THIS ONE**

D. `["5056 Donald Bren Hall", "Irvine", "CA 92697"]`

E. `["5056 Donald Bren Hall", "Irvine", "CA", "92697"]`

(b) (2 points) When we call `string_to_Address("5056 Donald Bren Hall, Irvine, CA  92697")`, what is the value assigned to `state_zip`?

A. `"CA  92697"`

B. `["CA  92697"]`

C. `["5056 Donald Bren Hall", " Irvine", " CA  92697"]`

D. `["CA", "92697"]` **# THIS ONE**

E. `[" CA", "  92697"]`

(c) (2 points) Below is an alternative way to write this function:

```
def string_to_Address2(entire_addr: str) -> Address:
    """ Create an Address with the contents of the string
    """
    comma_pos = entire_addr.find(",")
    street_addr = entire_addr[:comma_pos].strip()
    city_state_zip = entire_addr[comma_pos+1:]
    comma_pos = city_state_zip.find(",")
    city = city_state_zip[:comma_pos].strip()
    state_zip = city_state_zip[comma_pos+1:].split()
    state = state_zip[0]
    zip = state_zip[1]
    return Address(street_addr, city, state, zip)

assert string_to_Address2("5056 Donald Bren Hall, Irvine, CA  92697") == \
     Address("5056 Donald Bren Hall", "Irvine", "CA", "92697")
assert string_to_Address2("   24672 Raymond Way,   El Toro,CA 92630  ") == \
     Address("24672 Raymond Way", "El Toro", "CA", "92630")
```

When we call `string_to_Address2("5056 Donald Bren Hall, Irvine, CA  92697")`, what is the value assigned to `city_state_zip`?

A. `["Irvine", "CA", "92697"]`

B. `"5056 Donald Bren Hall, Irvine, CA  92697"`

C. `[" Irvine", " CA  92697"]`

D. `", Irvine, CA  92697"`

E. `" Irvine, CA  92697"` **# THIS ONE**

(d) (3 points) Below is a function:

```
def you_tell_me(s: str) -> str:
    """ You provide the docstring.
    """
    L = s.split()
    s = " ".join(L)
    return s
```

Which one of the following is the best (most accurate) docstring comment for this function?

A. Returns the same string it was passed.

B. Returns the parameter with multiple spaces between words reduced to one space. **#THIS**

C. Returns a list of the words in the parameter string.

D. Separates the parameter string into words and joins it back together again

E. Removes all spaces and punctuation from the parameter string and returns the result.

Problem 4 (15 points) **Topic: Lists of namedtuples containing lists**

(Continue using the definitions from Problem 1. As always, for full credit you should use functions previously defined on this exam where appropriate, rather than reinventing the wheel.)

(a) (3 points) Complete the definition of the function below, consistent with its header, docstring comment, and assertions, by filling in each blank with exactly one identifier, operator, or constant.

```
def Library_percentage_checked_out(Lib: Library) -> float:
    """ Return the percentage (0-100) of this library's books that are currently
         checked out.
    """  # SCORING: percentage_checked_out = 2, collection = 1
    return ____________________(Lib.____________________)  percentage_checked_out    collection

assert Library_percentage_checked_out(ElTo) == 1/3*100
assert Library_percentage_checked_out(IrvUP) == 0
```

(b) (6 points) Complete the definition of the function below, consistent with its header, docstring comment, and assertions, by filling in each blank with exactly one identifier, operator, or constant.

```
def Lib_with_most_checked_out(Libraries: 'list of Library') -> Library:
    """ Return the Library with the greatest percentage of checked-out books.
    """   # SCORING: 2 points each:   Libraries   Library_percentage_checked_out   0
    return sorted(____________________, key=____________________,

                    reverse = True)[____________________]
assert Lib_with_most_checked_out(OCPL) == Brea
```

(c) (6 points) Complete the definition of the function below, consistent with its header, docstring comment, and assertions, by filling in each blank with exactly one identifier, operator, or constant.

```
def oldest_book(Libraries: 'list of Library') -> Book:
    """ From all libraries in the list, return the one Book with the earliest
         publication year.  (Okay to assume there's one oldest book, no ties.)
    """
    oldest_book_so_far = None
    oldest_year_so_far = 9999   # This will be replaced by the first real year
    for L in ____________________: Libraries
        for bk in L.____________________: collection
            if bk.year < ____________________: oldest_year_so_far
                oldest_book_so_far = ____________________ bk
                oldest_year_so_far = ____________________ . ____________________ bk.year
    return oldest_book_so_far
assert oldest_book(OCPL) == bk9
```

Problem 5 (5 points) Topic: String formatting

(a) (3 points) Suppose we want to print library information in a table formatted as shown:

```
Pct. Out  Name                       Phone         Address
 33.333%  El Toro                    949-855-8173  24672 Raymond Way, El Toro, CA  92630
 66.667%  Brea                       714-671-1722  1 Civic Center Circle, Brea, CA  92821
  0.000%  Irvine University Park     949-786-4001  4152 Sandburg Way, Irvine, CA  92612
 50.000%  Tustin                     714-544-7725  345 East Main Street, Tustin, CA  92780
```

Given a Library, we can produce a formatted string with the `Lib_to_str` function below:

```
def Lib_to_str(Lib: Library) -> str:
    """ Return a formatted string for printing, including percentage checked out
    """
    format_string = —insert value from the choices below—
    return format_string.format(Library_percentage_checked_out(Lib), Lib.name,
                                Lib.phone, Lib.address)

assert Lib_to_str(ElTo) == \
" 33.333%  El Toro                    949-855-8173  24672 Raymond Way, El Toro, CA  92630"
```

Which of the following could we correctly assign to `format_string`? Circle *one or more* of A, B, C, D, or E; more than one may be correct.

A. `"{:7.3f}%  {:25s}{:12s}  {:1s}"` # THIS ONE

B. `"{7.3f}%  {25s}{12s}  {1s}"`

C. `"{:6.3f}%  {:25s}{:12s}  {:1s}"`

D. `"{:0.3f}%  {:25s}{:12s}  {:1s}"`

E. `"{1:7.3f}%  {2:25s}{3:12s}  {4:1s}"`

(b) (2 points) Which one of the following is the correct output from this print statement:

```
print("Buy one new for ${:3.2f}; call now, 1-800-555-5555.".format(23599.95))
```

A. Buy one new for $23,599.95; call now, 1-800-555-5555.

B. Buy one new for $9.95; call now, 1-800-555-5555.

C. Buy one new for $.95; call now, 1-800-555-5555.

D. Buy one new for $23599.95; call now, 1-800-555-5555. # THIS ONE

E. Buy one new for $ 23599.95; call now, 1-800-555-5555.

Problem 6 (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):

- 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 still working.