

Quiz 4

To get credit for this quiz, use the Quiz tool at eee.uci.edu to enter your answers, within the Sunday-to-Tuesday quiz period.

Problem 1 (5 points)

Match each code segment ((a) through (e)) with the output it produces (1 through 5). Use each alternative exactly once. Assume `L` has the value `[10, 20, 30, 40]`.

___ (a)	1.
<pre>for i in L: print(i)</pre>	30 40
___ (b)	2.
<pre>for i in range(len(L)): print(i)</pre>	10
___ (c)	3.
<pre>for i in range(len(L)): if L[i] > 20: print(L[i])</pre>	0 1 2 3
___ (d)	4.
<pre>for i in range(len(L)): if L[i] > 20: print(i)</pre>	10 20 30 40
___ (e)	5.
<pre>for i in L: if i < 20: print(i)</pre>	2 3

Problem 2 (4 points)

(a) For each sequence of Xs in the code below, indicate *the type of expression* (not the specific code) that belongs in that space. Choose from int, float, str, bool, Restaurant, or list.

```
def nonvowels(s: str) -> str:
    ''' Return a string containing the nonvowels in s
    '''
    result = ''
    for XXXXXXXXXXXX in s:
        if XXXXXXXXXXXX:
            XXXXXXXXXXXX += c
    return result
assert XXXXXXXXXXXX
assert nonvowels(XXXXXXXXXX) == ""
```

(b) For each sequence of Xs in the code below, indicate the *type of expression* (not the specific code) that belongs in that space. Choose from int, float, str, bool, Restaurant, or list.

```
def cheapest(L: list) -> Restaurant:
    ''' Return the Restaurant with the lowest price, from the parameter
    '''
    if L == []:
        return None
    XXXXXXXXXXXX.sort(key=Restaurant_price)
    return XXXXXXXXXXXX
assert cheapest(XXXXXXXXXX) == XXXXXXXXXXXX
assert cheapest([XXXXXXXXXX, R25, R26]) == R24
```

Problem 3 (11 points)

The cost of framing a picture depends on the material used for the frame and the size of the picture.

(a) (3 points) In the function below, fill in the blanks (with one string, number, operator, or identifier [variable/function/method name] per blank) according to the description given.

```
def price_per_inch(material: str) -> float:
    ''' If parameter is 'maple', return 2.95; if it's 'lacquer', return 3.50; if it's
        'aluminum', return 1.25; otherwise return 0
    '''
    if _____ == _____:
        return 2.95
    elif _____ == _____:
        return 3.50
    elif _____ == _____:
        return 1.25
    else:
        return _____

assert price_per_inch('bamboo') == 0
assert price_per_inch('lacquer') == 3.50
```

(b) (3 points) Below is a function header and docstring describing a task. First, write two `assert` statements that give examples of calling this function and the correct results for those examples. Then, complete the definition according to the docstring and parameter types given.

```
def material_needed(height: float, width: float) -> float:
    ''' Given the height and width of a picture (in inches), return
        the number of inches of framing material required (for all
        four sides of the picture, of course).
    '''
```

(c) (5 points) For the function described below, first write two `assert` statements as examples/tests of its operation, then complete the function definition according to the description. Where appropriate, call the functions you defined above.

```
def frame_cost(height: float, width: float, material: str) -> float:
    ''' Return the cost of framing a picture whose size and framing
        material are specified by the parameters
    '''
```

Problem 4 (10 points)

Suppose we have this definition:

```
Date = namedtuple('Date', 'year month day')
```

where all three fields are numbers, so that February 14, 2014 would be represented as `Date(2014, 1, 14)`. [You might think about how to define a function to convert a `mmddyy` string to a `Date`, or to convert a `Date` to a `MonthDayYear` string, but neither one is part of this quiz.]

Suppose we also have this definition:

```
DrivingRecord = namedtuple('DrivingRecord', 'name license age tickets')
```

where `name` is a string representing a driver's name, `license` is a string representing his or her driver's license number, `age` is the driver's age, and `tickets` is a (possibly empty) list of `Date` objects containing the dates on which the driver has received a traffic ticket (i.e., was cited by a police officer for violating a driving law).

Complete the two function definitions below, consistent with their headers, docstrings, and assertions.

```
def is_dangerous(d: DrivingRecord, limit: int) -> bool:
    ''' Return True if d has more tickets than the limit (and false otherwise)
    '''
```

```
def dangerous_drivers(DRL: 'list of DrivingRecord', limit: int) -> 'list of str':
    ''' Return a list of the names of drivers in DRL who have
        more tickets than the specified limit. '''
```

```
assert dangerous_drivers(
    [DrivingRecord('a', '1', '16', [Date(1,2,3),Date(2,3,4),Date(3,4,5)]),
     DrivingRecord('b', '1', '66', [Date(1,2,3),Date(2,3,4)]),
     DrivingRecord('c', '1', '66', [Date(1,2,3)]),
     DrivingRecord('d', '1', '116', []),
     DrivingRecord('e', '1', '116', [Date(1,2,3),Date(2,3,4),Date(3,4,5)])], 1)
== ['a', 'b', 'e']
```

Problem 5 (5 points)

Below is some code to draw a variety of figures using `tkinter`, as we did in the lab. However, you don't need to know any `tkinter` operations to answer this question. (Some function definitions below are missing or incomplete. None of the missing details is necessary to answer this question.)

```
def draw_them_all():
    ''' Draw a variety of figures. The question asks you to
        count the total number of eyes drawn.
    '''
    draw_eye(200,200)
    draw_Martian_face(0,0)
    draw_Martian_face(100,100)

def draw_Martian_face(x: int, y: int):
    ''' Draw, centered at (x, y), a Martian face with three eyes.
    '''
    draw_head(x, y)
    draw_nose(x, y)
    draw_mouth(x, y-50)
    draw_eye(x-50, y+20)
    draw_eye(x, y+20)
    draw_eye(x+50, y+20)

def draw_eye(x: int; y: int) -> None:
    ''' Draw an eye centered at (x, y)
    '''
    [body omitted]

draw_them_all()
```

(a) (3 points) If we call the function `draw_them_all` once, how many times in total does an eye get drawn?

(b) (2 points) Circle (or indicate some other way) the *last* call to `draw_eye` that gets executed in a complete call to `draw_them_all`.