Quiz 9

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 (8 points)

Suppose an exam has two problems, each worth 20 points. We want to see how students' scores on Problem 1 relate to their scores on Problem 2, so we decide to make a scatter plot.

Our students come in a list of Score records, defined as follows:

```
from collections import namedtuple
Score = namedtuple('Score', 'p1 p2')
TOPSCORE = 20
```

Both `p1` and `p2` are ints between 0 and `TOPSCORE` (inclusive). With this list:

```
scorelist = [Score(p1=0, p2=0),
             Score(p1=1, p2=1),
             Score(p1=1, p2=5),
             Score(p1=4, p2=2),
             Score(p1=5, p2=0)]
```

the scatter plot of scores would look like this (except that we've omitted the 75% of the table that would show scores greater than 5):

```
5| *
4|
3|
2|    *
1| *
0|*    *
  ------
  012345
```

To keep things simpler in our problem, we're going to omit the axes and the labels and print just the 21-by-21 body of the plot.

```
# Initialize the table to a 20-by-20 table of blanks
table = [ ]
for row in range(TOPSCORE+1):
    table_row = [ ]
    for col in range(TOPSCORE+1):
        table_row.append(' ')
    table.append(table_row)

# Populate the table with an asterisk for each student's two scores.  (When two
# students have the same pair of scores, just one asterisk appears.)
for s in scorelist:
    table[s.p2][s.p1] = '*'

# Print the 20-by-20 table
for row in range(TOPSCORE,-1,-1):
    for col in range(TOPSCORE+1):
        print(table[row][col], sep='', end='')
    print() # Print the default end= character, a newline
```

Please answer each of the following questions in just a few English words:

A. Why do we have to say `range(TOPSCORE+1)`?

B. Why do we have to say `table[s.p2][s.p1]` and not `table[s.p1][s.p2]`?

C. When we print the table, why do we print the rows in a backwards range (`TOPSCORE` down to 0)?

D. Why do we have `sep=''` and `end=''` when we print a row of the table?

Problem 2 (17 points)

Suppose we have a list of names, some of which may occur more than once on the list. For example:

```
NL = ['Joe', 'Sam', 'Joe', 'Jill', 'Joe', 'Joe', 'Jill', 'Sam', 'Jane', 'Jane', 'Jane', 'Joe', 'John']
```

And suppose that we want to know which name occurs most frequently.

(a) (5 points) First we can create a dictionary that gives us a collection of each distinct name on the list, along with the number of times it occurs. Fill in the blanks of the following definition, with one identifier, constant, or operator in each blank, to be consistent with the provided code:

```
def tally_names(L: [str]) -> dict:
    ''' Return a dictionary with each unique string in L as the key and
        the number of times that string occurs in L as the value.
    '''
    result = { }
    for s in __________:
        if __________ in __________:
             __________ [__________] __________  __________
        else:
             __________ [__________] __________  __________
    return __________

assert tally_names(NL) == {'Sam': 2, 'Jill': 2, 'Joe': 5, 'Jane': 3, 'John': 1}
```

(b) (2 points) We want to find the most frequently occurring name, but we can't sort the dictionary because dictionaries, as we know, are inherently unsorted (they can't be, because of how they're built [using a "hash table," the details of which are a topic for ICS 33 or ICS 46]). But we can use a function like the one below to convert the dictionary to a list of key-value pairs (where each pair is a two-item list, [key, value]).

```
def dict_to_list(d: dict) -> 'list of [key, value] pairs':
    ''' Convert dictionary (with key/value entries) to a list of [key, value] pairs
    '''
    result = [ ]
    for key in d:
        result.append([key, d[key]])
    return result
```

What does the following statement print, using the definition of NL above? (Hint: Look at the assertion for the previous part.)

```
print(dict_to_list(tally_names(NL)))
```

(c) (5 points) The following sequence of statements prints the most frequently occurring string in the original list `NL`, along with the number of times it occurs. Complete each statement below to produce this result, supplying one identifier, operator, or constant for each blank. (Hint: The problem contains many clues.)

```
def second_item(L: list) -> 'any':
    ''' Return second field (L[1]) of a list, to use with key= in Sort() method
    '''
    return __________ [ __________ ]

list_of_string_frequency_pairs = __________ (__________(NL))
__________ . __________(key=__________, reverse=True)
most_frequent_pair = __________ [ __________ ]
print("The string '", __________ [ __________ ], "' occurs ",
     __________ [ __________ ], ' times.', sep='')
```

(d) (1 point) What data structure (chosen from dictionaries, sets, or tuples) could we use (instead of the two-item key-value list) to represent each word with its frequency? (One word.)

(e) (1 point) The result of `tally_names(NL)` in part (a) above is a dictionary with strings (names) as the keys and numbers (frequency counts) as the values.

Describe in just a few English words the meaning of `list(tally_names(NL).keys())`.

(f) (3 points) Consider this code:

```
def remove_duplicates(L: [str]) -> [str]:
    ''' Return a list containing the strings in L with no duplicates
    '''
    result = set()
    for s in L:
        result.add(s)
    return list(result)
```

(f.1) What data structure (chosen from dictionaries, sets, and tuples) does this code use? (One word.)

(f.2) Why does this code call `list()` in the `return` statement?

(f.3) Would you expect this assertion to pass (i.e., be True)?

```
assert sorted(remove_duplicates(NL)) == sorted(list(tally_names(NL).keys()))
```