Selected Practice Questions

Understanding Code (**DO NOT RUN THEM ON IDLE. DO THIS QUESTION ON PAPER)**

For each of the code below, write what the output(s) will be. (Taken from CS1010S 2014 Finals)

A.

```
x = [1,2,3]
if x is [1,2,3]:
    x = [x] + [4,5]
if len(x)<=3:
    y = [x] + [6,7]
elif len(x)<=5:
    y = [x] + [10,11]
print(y)
```

B.

```
x = (1,2,3,4)
a = x[1::2]
if not a:
    print(a)
else:
    print(a+x)
```

C.

```
a = 0
for i in range(2,7):
    if i%4==0:
        a/=2
        i+=2
    else:
        a+=i
        i+=1
print(a)
```

D.

```
x = 1
y = 5
def foo(y):
    y = 10
    def bar(z):
        return x+y+z
    return bar(y)
print(foo(x))
```

E.

```
x=[0,1]
def test(x):
    try:
        for i in range(2,6):
            x = x.append(i)
    except:
        print("Bad things!")
    else:
        print(x)
    finally:
        print("Finished!")
test(x)
```

F.

```
x = [1,3,2]
def foo(x):
    x.sort()
    x = x + [4,5]
    x.extend([6,7])
foo(x)
print(x)
```

Recursion & Iteration

Q1 (Taken from CS1010S 2014 Midterm)

****DO THIS QUESTION ON PAPER****

Consider the following alternating series s_{11}:

$$s_{11}(n) = 1 - 2 + 3 - 4 + \cdots n$$

A. **[Warm Up]** Write an **recursive** function `s11(n)` that returns the value for $s_{11}(n)$. [4 marks]

C. Write an **iterative** function `s11(n)` that returns the value for $s_{11}(n)$. [4 marks]

Now, consider the following alternating series s_{ij} where $i, j \geq 1$:

$$\begin{aligned} s_{21}(n) &= 1 + 2 - 3 + 4 + 5 - 6 + 7 + 8 - \cdots n \\ s_{12}(n) &= 1 - 2 - 3 + 4 - 5 - 6 + 7 - 8 - \cdots n \\ s_{31}(n) &= 1 + 2 + 3 - 4 + 5 + 6 + 7 - 8 + \cdots n \end{aligned}$$

Note that the subscript i denotes the number of terms with a positive sign and the subscript j denotes the number of terms with a negative sign. Basically these series will alternate between i positive terms and j negative terms.

E. Write a function `s21(n)` that returns the value for $s_{21}(n)$. [5 marks]

F. Write a function `make_s(i,j)` that returns the function $s_{ij}(n)$. In other words, we could have defined `s11(n)` for Part (A) (or (C) depending on your implementation) as follows:

```
s11 = make_s(1,1)
s12 = make_s(1,2)
```

[5 marks]

List Manipulation

Q1

Transposing is the action where a matrices rows and columns are swapped. Eg:

$$A\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix} \quad A^T\begin{bmatrix}1 & 4 & 7\\ 2 & 5 & 8\\ 3 & 6 & 9\end{bmatrix}$$

$$A\begin{bmatrix}1 & 4 & 3\\ 8 & 2 & 6\\ 7 & 8 & 3\\ 4 & 9 & 6\\ 7 & 8 & 1\end{bmatrix} \quad A^T\begin{bmatrix}1 & 8 & 7 & 4 & 7\\ 4 & 2 & 8 & 9 & 8\\ 3 & 6 & 3 & 6 & 1\end{bmatrix}$$

Write a function **transpose_matrix** to transpose **any** given 2d matrix.

Q2 *(Taken from CS1010S PE 2014)*

Problem 1 Match and Sieve (10 marks)

a) [6 marks] Write a function `match` that takes in an item (which can be a string or scalar object), and a sequence (which can be a tuple or a list object) and returns a tuple of 2 elements: a result list and a count: If the item matches one or more elements in the sequence, the matched elements are collected into the result list, the count indicates the number of matched items in the result list. Some sample runs are show below:

```
>>> match(2, [1, 3, 4, 1, 5, 2])
([2], 1)

>>> match(5, (-2, 3, -5, 1, 5, 5, 'a', 5, 8, 3.0))
([5, 5, 5], 3)

>>> match(3.0, (-2, 3, -5, 1, 3.0, 5, 'a', 5, 8, 3.0))
([3, 3.0, 3.0], 3)

>>> match(0, (-2, 3, -5, 1, 5, 5, 'a', 5, 8, 3.0))
([], 0)

>>> match(-22, [-2, 3, -22, 1, 5, 5, 'a', 5, -22, 3.0])
([-22, -22], 2)

>>> match('a', [-2, 3, -22, 'hello', 5, 5, 'a', 5, -22, 3.0, 'a'])
(['a', 'a'], 2)
```

List & Dictionary Manipulation

Q1 *(Taken from CS1010S 2014 Tutorial 7)*

Ignoring punctuation, an English sentence is a collection of words, and each word is a collection of letters. Suppose we would like to (for the fun of it) create a list representation of an English sentence. We could represent each word as a list of letters (for example, the word 'cat' would be represented as `['c', 'a', 't']`). A sentence would then be a list of such word representations. Some examples :

```
"CS1010S Rocks"
[['C', 'S', '1', '0', '1', '0', 'S'], ['R', 'o', 'c', 'k', 's']]

"Python is cool"
[['P', 'y', 't', 'h', 'o', 'n'], ['i', 's'], ['c', 'o', 'o', 'l']]
```

a) Write a function **`count_sentence`** that takes a sentence representation (as described above) and returns a series list with two elements: the number of words in the sentence, and the number of letters in the sentence.

Assume that spaces count as 1 letter per space, and that there is exactly 1 space between each word (but none at the start or end of the sentence).

b) Write a function **`letter_count`** that takes a sentence and returns a list of lists, where you have one list for each distinct letter in the sentence and each list has two elements.

The first element of the list pair is the letter and the second element is the count for the letter. The order of the list pairs does not matter.

Example execution:

```
>>> letter_count([['s', 'h', 'e'], ['l', 'i', 'k', 'e', 's'],
                  ['p', 'i', 'e', 's']])
[['s', 3], ['h', 1], ['e', 3], ['l', 1], ['i', 2], ['k', 1], ['p', 1]]
```

c) Write a function **`most_frequent_letters`** that takes a sentence representation and returns a list of letters that occur most frequently in the given sentence.

The order of the letters does not matter. If only one such letter exists, then return a list with one element. If the sentence is empty, return an empty list.

Example execution:

```
>>> most_frequent_letters([['s', 'h', 'e'], ['l', 'i', 'k', 'e', 's'],
                           ['p', 'i', 'e', 's']])
['e', 's']
```

Q2 *Taken from CS1010S PE 2014)*

b) [4 marks] Write a function `sieve` that takes in an item (which can be a string or scalar object), and a sequence of pairs (the sequence can be a tuple or a list object, with tuples or lists as elements) and returns a dictionary object: All the pairs in the input sequence are transformed into key-value pairs in the dictionary, except for the pair with a key that matches the item, if any, which will not be included in the dictionary. You may assume that all the inputs are well-formed, i.e., the input item and the sequence of pairs are of the types as specified above, and that there are no duplicate keys. Some sample runs are show below:

```
>>> sieve(2, (('a', 4), ('b', 'b'), ('c', 2), ('d', 2), (2, 'e')))
{'a': 4, 'c': 2, 'b': 'b', 'd': 2}

>>> sieve('b', (('a', 4), ('b', 'b'), ('c', 2), ('d', 2), (2,'e')))
{'a': 4, 2: 'e', 'c': 2, 'd': 2}

>>> sieve((2, 3), \
          (('a', 4), ('b', (2, 3)), ('c', 2), ('d', (2, 3)), ((2, 3),'e')))
{'a': 4, 'c': 2, 'b': (2, 3), 'd': (2, 3)}

>>> sieve('c', \
          (('a', 4), ('b', [2, 3]), ('c', 2), ('d', (2, 3)), ((2, 3), 'e')))
{'a': 4, 'b': [2, 3], (2, 3): 'e', 'd': (2, 3)}

>>> sieve(3.0, [('a', 4), ('b', [2, 3]), ('c', 2), (3, (2, 3)), (2, 'e')])
{'a': 4, 'c': 2, 'b': [2, 3], 2: 'e'}

>>> sieve('f', [('a', 4), ('b', [2, 3]), ('c', 2), ('d', (2, 3)), (2, 'e')])
{'a': 4, 2: 'e', 'c': 2, 'b': [2, 3], 'd': (2, 3)}
```

Logic

Q1 *(Taken from https://open.kattis.com/contests/c5pjur/problems/conundrum)*

There is a random string on a whiteboard. Tom is bored and decided to do a puzzle. Every day he will erase one letter of the text and replace it with a different letter, so that, in the end, the whole text reads “TomTomTomTom”. Since Tom will change one letter each day, he hopes that people will not notice.

Tom would like to know how many days it will take to transform a given text into a text only containing his name, assuming **he substitutes one letter each day.** You may assume that the length of the original text he writes is a multiple of 33.

For simplicity, you can ignore the case of the letters, and instead assume that all letters are

upper-case.

Input
The first and only line of input contains the cipher text on the whiteboard. It consists of at most 300300 upper-case characters, and its length is a multiple of 33.

Output
Output the number of days needed to change the cipher text to a string containing only Per's name.

Sample Input 1	Sample Output 1
TIMTAM	2
TOILET	4

Explanation for input 1
TIMTAM → TOMTAM → TOMTOM (2)
Explanation for input 2
TOILET → TOMLET → TOMTET → TOMTOT → TOMTOM (4)

Q2

You are given two sorted lists. Can you write a function to find what is their combined median?

Eg,

l1 = [12,46,67,90]
l2 = [13,56,89,95,200]

```
In [32]: l1
Out[32]: [12, 46, 67, 90]

In [33]: l2
Out[33]: [13, 56, 89, 95, 200]

In [34]: find_combined_median(l1,l2)
Out[34]: 67
```

Note: Be warned, how do you find the median of a list with total even number of elements?

Image Manipulation

For this, the Labs, mock PE, and PE itself is the best practice that you can do. Anything else might not be of appropriate difficulty / make sense to you.