CSCI-UA.0380 - Midterm Practice Questions

Part 1 - Short answer sample questions. (These questions show the *type* of questions that may be on the exam, but they do not indicate the exact topics, length or difficulty level of the exam questions).

1. **Vigenère cipher, Substitution Cipher** – answer the questions below...

   (a) You just received the following message encrypted using the Vigenère cipher: `USYK`

   Decrypt the message using the key , `TOY`, and the Vigenère square on the right. Show your work for partial credit.

```
          Key: TOYT
Cipher Text: USYK



     plain text --> BEAR
```

	ABCDEFGHIJKLMNOPQRSTUVWXYZ
A	ABCDEFGHIJKLMNOPQRSTUVWXYZ
B	BCDEFGHIJKLMNOPQRSTUVWXYZA
C	CDEFGHIJKLMNOPQRSTUVWXYZAB
D	DEFGHIJKLMNOPQRSTUVWXYZABC
E	EFGHIJKLMNOPQRSTUVWXYZABCD
F	FGHIJKLMNOPQRSTUVWXYZABCDE
G	GHIJKLMNOPQRSTUVWXYZABCDEF
H	HIJKLMNOPQRSTUVWXYZABCDEFG
I	IJKLMNOPQRSTUVWXYZABCDEFGH
J	JKLMNOPQRSTUVWXYZABCDEFGHI
K	KLMNOPQRSTUVWXYZABCDEFGHIJ
L	LMNOPQRSTUVWXYZABCDEFGHIJK
M	MNOPQRSTUVWXYZABCDEFGHIJKL
N	NOPQRSTUVWXYZABCDEFGHIJKLM
O	OPQRSTUVWXYZABCDEFGHIJKLMN
P	PQRSTUVWXYZABCDEFGHIJKLMNO
Q	QRSTUVWXYZABCDEFGHIJKLMNOP
R	RSTUVWXYZABCDEFGHIJKLMNOPQ
S	STUVWXYZABCDEFGHIJKLMNOPQR
T	TUVWXYZABCDEFGHIJKLMNOPQRS
U	UVWXYZABCDEFGHIJKLMNOPQRST
V	VWXYZABCDEFGHIJKLMNOPQRSTU
W	WXYZABCDEFGHIJKLMNOPQRSTUV
X	XYZABCDEFGHIJKLMNOPQRSTUVW
Y	YZABCDEFGHIJKLMNOPQRSTUVWX
Z	ZABCDEFGHIJKLMNOPQRSTUVWXY

   (b) The Vigenère cipher is an improvement over the regular substitution cipher. Without knowing the key for a substitution cipher, what's one way of determining what key was used to encrypt the message (and consequently *breaking* the encryption)?

Simple letter frequency analysis of cipher text from substitution cipher will reveal mapping / original key. (Though, as someone pointed out in class, Vigenère can also be cracked!)

   (c) Complete the following definition for decrypting a message encrypted with a simple substitution cipher. Assume that the methods, generate_key and gen_consecutive_chars both exist.

```
"""
:param password: (str) the password used to generate a key
:param ciphertext: (str) the text to be decrypted
:return: (str) the plain text that results from decrypting the ciphertext
"""
def sub_decrypt(password, ciphertext):
    key = gen_key(password)
    alpha = gen_consecutive_chars()
    # finish this implementation
    plain_text = ''
    for ch in ciphertext:
        try:
            plain_text += alpha[key.index(ch)]
        except ValueError:
            plain_text += ch

    return plain_text
```

2. **Functions, Classes** – answer the questions in the right column.

```
def add_nums(x, y, z):
    return x + y + z

def make_new_func(func, arg):

    def new_func(a, b):
        return func(arg, a, b)

    return new_func

new_add_1 = make_new_func(add_nums, 10)
new_add_2 = make_new_func(add_nums, 23)

print(new_add_1(5, 7))
print(new_add_2(5, 7))
```

What is the output of the code on the left – error and/or no output is possible. If there's an error, specify where the error is.

```
22
35
```

```
def pluralize(word):
    return word + 's'

def inquire(word):
    return word + '?'

new_func = compose(inquire, pluralize)
result = new_func('bee')
print(result) # bees?
```

The code on the left prints out:

```
bees?
```

Fill in the missing line in the code below so that the implementation of the function `compose` results in the specified output above.

```
def compose(f, g):
    def new_f(a):

        return f(g(a))

    return new_f
```

```
def foo():
    print('calling foo')

def baz():
    print('calling baz')

def qux():
    print('calling qux')

d = {"f":foo, "b":baz, "q":qux}

for ch in 'baffling quiz':
    try:
        call_it = d[ch]
        call_it()
    except KeyError:
        continue
```

What is the output of the code on the left – error and/or no output is possible. If there's an error, specify where the error is.

```
calling baz
calling foo
calling foo
calling qux
```

```
class Rectangle():

    def __init__(self, width, height):
        self.w = width
        self.h = height



r = Rectangle(5, 15)
perimeter = r.perimeter()
print(perimeter) # outputs 40
```

Add a method called `perimeter` to the class on the left (you can write your method definition in this box). It should work as shown in the code below the class definition.

```
def perimeter(self):
    return self.w * 2 + self.h * 2
```

Part 2 – coding questions. (These questions show the *type* of questions that may be on the exam, but they do not indicate the exact topics, length or difficulty level of the exam questions).

3. Write a program that reads in a file that contains the names and the amounts of money each person spent on lunch every day for 5 days. The program should print out the name of the person that spent the most money for the 5 day period, along with the amount that they spent.

   The data in the file contains a header specifying what data is in the file. Each person is represented by a row in the file. Their name and the amount of money the spent each day is separated by pipe (vertical line) characters. Here's an example of the file format.

```
name|day1|day2|day3|day4|day5
alice|$8.00|$7.50|$9.97|$8.00|$6.25
bob|$2.99|$5.50|$5.50|$5.50|$2.99
carol|$7.50|$6.99|$5.00|$10.00|$5.25
daniel|$9.25|$10.99|$9.25|$15.00|$12.00
eve|$7.50|$7.00|$8.00|$8.00|$7.00
```

   Assume the the file you're working with is called lunch.txt, and it's in the same directory that you're running your program from.

```
d = {}
with open('lunch.txt', 'r') as f:
    for line in f:
        if 'name' in line:
            continue
        parts = line.split('|')
        d[parts[0]] = 0
        for i in range(1, len(parts)):
            d[parts[0]] += float(parts[i][1:])

who = max(d, key=d.get)
print(who, '${}'.format(d[who]))
```

4. **Image Processing.**

Original image:

Half of image mirrored vertically (along horizontal axis):

Image inverted (upside-down):

Write two functions to create the mirrored and inverted images above (2nd and 3rd columns). Both functions must change the original image (you can't use Image.new). See below for example usage:

```
my_img = Image.open('raccoon.jpg')
mirror_half(my_img)
my_img.show()
```

```
my_img = Image.open('raccoon.jpg')
upside_down(my_img)
my_img.show()
```

```
import math
def mirror_half(img):
    pixels = img.load()
    half = math.ceil(img.size[1] / 2)
    for x in range(img.size[0]):
        other_y = img.size[1] // 2
        for y in range(half, img.size[1]):
            pixels[x, y] = pixels[x, other_y]
            other_y -= 1


def upside_down(img):
    pixels = img.load()
    half = img.size[1] // 2
    for x in range(img.size[0]):
        for y in range(half):
            other_y = img.size[1] - y - 1
            pixels[x, y], pixels[x, other_y] = pixels[x, other_y], pixels[x, y]
```

5. **Working with Lists -** use the following list of dictionaries to answer the questions below. Each dictionary represents a moon:

`planet` - the planet that the moon orbits
`num` - the number of the moon based on planet
`name` - the moon's name
`r` - the moon's radius in kilometers
`m` - the moon's mass in kilograms * 10^{21}

```
moons = [
    {'planet':'Jupiter', 'num':3, 'name':'Ganymede', 'r':2634, 'm':148.2},
    {'planet':'Saturn', 'num':6, 'name':'Titan', 'r':2576, 'm':134.5},
    {'planet':'Jupiter', 'num':4, 'name':'Callisto', 'r':2410, 'm':107.6},
    {'planet':'Jupiter', 'num':1, 'name':'Io', 'r':1821, 'm':89.3},
    {'planet':'Jupiter', 'num':2, 'name':'Europa', 'r':1560, 'm':48},
    {'planet':'Neptune', 'num':1, 'name':'Triton', 'r':1353, 'm':21.5},
    {'planet':'Uranus', 'num':3, 'name':'Titania', 'r':788, 'm':3.5}]

GRAVITY = 6.67E-11
```

Determine which moons have a gravity that's greater than `1m/s²` ... and create a list of those moons using their planet, number and name together as a single string.

Gravity is calculated by using the formula: `G * M / r2` ... where G is the Gravitational constant (listed in the code above as the variable, GRAVITY), M is the mass in kg and r is the radius in km. Note that you'll have to multiply the mass in the dictionary by 10^{21} and the distance by 10^{3} to get the correct orders of magnitude.

A moons's name consists of its planet, number, and *actual* name. For example: `'Jupiter 1 – Io'`. The resulting list should be similar to: `['Jupiter 3 - Ganymede', 'Saturn 6 - Titan' ...]`

(a) Use a regular for loop to implement this.

```
// helper functions

def get_gravity(d):
    m = d['m'] * 10 ** 21
    r = d['r'] * 10 ** 3
    return (GRAVITY * m) / r ** 2

def get_name(d):
    planet, num, name = d['planet'], d['num'], d['name']
    return '{} {} - {}'.format(planet, num, name)

new_list = []
for d in moons:
    if get_gravity(d) > 1:
        new_list.append(get_name(d))
```

(b) Use list comprehensions to implement this (you can write as many helper functions as you need).

```
// using above helper functions
print([get_name(d) for d in moons if get_gravity(d) > 1])
```

(c) Use both filter and map to implement this (you can write as many helper functions as you need).

```
// using above helper functions
filtered = filter(lambda d: get_gravity(d) > 1, moons)
mapped = list(map(lambda d: get_name(d), filtered))
print(mapped)
```