Note: These questions are not a comprehensive study guide! They are given here to provide you with a sense of the types of questions that may be on the second midterm.

To prepare thoroughly for the second midterm you should review readings from the text, class notes, lab notes, projects and project solutions.

The format of W17 midterm2-part1 (in class) will be multiple choice questions (Scantron). The format of W17 midterm2-part2 (in all week 8 labs) will be similar to projects, with .py files developed according to a project specification and CIS 210 style guidelines, using the IDLE programming environment (only), and submitted via Canvas.

For numbers (1) through (5), replace ??s with the expected results.

(1) [4 pts.] Given:

def q1(slist):
    '''(list of str) -> ??'''
    slen = len(slist)
    sum = 0
    for s in slist:
        sum += s.count('x')
    avg = sum / slen
    return avg

li = ["CIS 2xx", "name", "xxxx", "CIS 3xx"]

>>> type(q1(li))
float

>>> q1(li)
2.0
(2) [4 pts.] Given

def q2(w):
    '''(str) -> Boolean'''
    if len(w) <= 1:
        return True
    elif w[0] != w[-1]:
        return False
    else:
        return q2(w[1:len(w)-1])

>>> q2('abcd')
False

>>> q2('reverserever')
True

(3) [4 pts.] Given:

def q3(x, y):
    '''(int, int) -> None'''
    x = f(x, y)
    y = f(y, x)
    print(x, y)
    return None

def f(x, y):
    '''(int, int) -> int'''
    x = 2 * x
    y = 2 * y
    if y > x:
        return y - x
    else:
        return x - y

>>> q3(20, 5)
30 50
Given:
def q4(li, div):
    ''' (list of ints, int) -> list of two lists of ints'''
    lowtodiv = []
    higherdiv = []
    for item in li:
        if item <= div:
            lowtodiv.append(item)
        else:
            higherdiv.append(item)
    return [lowtodiv, higherdiv]

>>> q4([1, 3, 5, 7, 9, 11, 13, 15], 10)
[[1, 3, 5, 7, 9], [11, 13, 15]]
>>> q4([97, 98, 99], 99)
[[97, 98, 99], []]

(5) [4 pts.] Given:

def q5(ch):
    '''(str) -> Boolean'''
    barD = {
        '0':'11000',
        '1':'00011',
        '2':'00101',
        '3':'00110',
        '4':'01001',
        '5':'01010',
        '6':'01100',
        '7':'10001',
        '8':'10010',
        '9':'10100'}
    bar = barD[ch]
    return bar[0] == '1'
>>> q5('5')
False
>>> q5('8')
True

(6a) [4 pts.] Replace ??s with the expected results. If the result is an error, write [ERROR].

```python
def mean(li):
    '''(list of ints) -> float
    returns average of the integer
    values in li.

    >>> mean([1, 10, 4, 1])
    4.0
    ...'''
    sum = 0
    for item in li:
        sm = sum + item  # sm should be sum - logic
    avg = sum / len(li)
    return avg

def q6(li):
    '''(list of ints) -> None

    Reports average value of items in li.

    >>> q6([1, 10, 4, 1])
    Average is 4.0
    >>> q6([])
    Empty list
    ...'''
```
if len(li) >= 0:# should be > 0 – runtime error
    print('Average is', mean(li))
else:
    print('Empty list')
return None

>>> q6([1, 10, 4, 1])
Average is 0.0
>>> q6([])
Average is ['ERROR']

(6b) [4 pts.] For each of the two bugs,
-- circle the bug
-- indicate whether it is a syntax, runtime, logic, or documentation error
-- fix the bug

(7) [8 pts.] Complete the docstring for function q7. Include three example function calls that are also test cases for three different equivalence classes for possible input or output values. Mention the specific equivalence class in comments next to the example function calls.

def q7(myl, item):
    '''(list, object) -> Boolean

    returns True is item is a member
    of myl, otherwise return False
    ...'''

    >>> q7([1, 2, 3], 3) #returns True
    True
    >>> q7([1, 2, 3], 4) #returns False
    False
    >>> q7([], 99) #myl is empty list
    False
    ...
for checkitem in myl:
    if checkitem == item:
        return True

return False

(8) [8 pts.] Replace the ??s with the results of executing the following code in the Python Shell.

```python
>>> states = ['OR', 'WA', 2]
>>> state = 'ID'
>>> states.insert(0, state)
>>> states[-1] += 1
>>> states
['ID', 'OR', 'WA', 3]

>>> state = 'MT'
>>> states[-1] += 1
>>> states = states.insert(0, state)
>>> print(states)
None

>>> state = 'ca'
>>> state.upper()
>>> state
'ca'

>>> state = state.upper ()
>>> state
'CA'
```