CIS 210 Winter 2019 Midterm Example Questions

Note: These questions are not a comprehensive study guide! They are given here to provide a sense of the types of questions that may be on the CIS 210 midterm exam.

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

The midterm exam will be given in-class, and will comprise multiple choice questions, short-answer questions, and questions where the solution will require you to write Python code according to the usual CIS 210 style guidelines.

No outside resources are allowed during the exam, with the exception of one index card of handwritten notes.

```python
>>> isinstance(101, float) == True
??

>>> isinstance(101, float)
??
```

Given the following Python code:

```python
>>> x = 'CIS 210'
>>> id(x) 439150
>>> y = x
>>> id(y)
??-1
>>> x = 'the end'
>>> id(x)
??-2
>>> y
??-3
```

4391509160 refers to a(n)

a) assignment statement b) function c) None type d) keyword e) memory location

The value printed at ??-1 will also be 4391509160 (yes or no); the value printed at ??-2 will also be 4391509160 (yes or no).

a) yes/yes b) no/no c) yes/no d) no/yes

The value printed at ??-3 will be

a) 4391509160 b) 'CIS 210' c) 'the end' d) None

The following Python code

```python
>>> x = 'hi'
>>> x = 0
>>> x = x < 0
```
demonstrates which Python characteristic?

a) strong typing  
b) dynamic typing  
c) operator overloading  
d) loops  
e) conditionals  

The following Python code

```python
>>> x = 'hi' + '-' + 'bye'
>>> y = 99 + 100
```

demonstrates which Python characteristic?

a) strong typing  
b) dynamic typing  
c) operator overloading  
d) weak typing  
e) static typing  

The following Python code

```python
>>> x = 'hi' + 99
Traceback (most recent call last):
  File "<pyshell#41>", line 1, in <module>
    x = 'hi' + 99
TypeError: must be str, not int
```

demonstrates which Python characteristic?

a) strong typing  
b) dynamic typing  
c) operator overloading  
d) weak typing  
e) static typing  

What will be printed when the following Python code is executed?

```python
n = 5
mysum = 0
for ctr in range(1, n):
    myctr = mysum + ctr
print(mysum)
```  

??

This code does not work as intended. This bug may be attributed to Python’s

a) strong typing  
b) dynamic typing  
c) operator overloading  
d) weak typing  
e) static typing  

What will be printed when the following Python code is executed?

```python
n = 5
mysum = 0
for ctr in range(1, n):
    mysum = mysum + ctr
print(mysum)
```

This code is an example of

a) accumulator pattern  b) TypeError  c) conditional  d) indefinite iteration  e) infinite loop

Given the following Python code:

```python
import math

def isInCircle(x, y, r):
    '''(number, number, number) -> ??
    Returns True if point (x, y) is in the circle with radius r.
    >>> isInCircle(0, 0, 1)
    True
    >>> isInCircle(.5, .5, 1)
    True
    >>> isInCircle(1, 2, 1)
    False
    '''
    d = math.sqrt(x**2 + y**2)
    isIn = d <= r
    return isIn
```

Complete the type contract: ??

Which code would give the same results as `isInCircle` lines 12-14 (changes are in bold)?

a) `d = math.sqrt(x**2 + y**2)
   return d = r`
b) `d = math.sqrt(pow(x, 2) + pow(y, 2))
   return d <= r`
c) `d = math.sqrt(x**2 + y**2)
   return d < r`
d) `d = math.sqrt(pow(x, 2) + pow(y, 2))
   isIn = d < r
   return isIn`
e) `d = math.sqrt(x**2 + y**2)
   return d`
Given the following Python code:

```python
1 - >>> ftemp = 212
2 - >>> ctemp = (ftemp - 32) * 5/9
3 - >>> ctemp = ftemp - 32 * 5/9
```

The value of `ctemp` will [??] from line 2 to line 3; the type of `ctemp` will [??] from line 2 to line 3.

a) stay the same/change  
b) change/stay the same  
c) stay the same/stay the same  
d) change/change  

Given the following Python code:

```python
def q29(s1):
    '''(str) -> str

        s2 = ''
        for ch in s1:
            if ch not in s2:
                s2 += ch

        return s2
```

Which brief description is appropriate for `q29`?

a) copies `s1` to `s2`; returns `s2`  
b) copies all characters except the last character in `s1` to `s2`; returns `s2`  
c) copies 1st occurrence of each character in `s1` to `s2`; returns `s2`  
d) determines whether `s1` is an empty string  
e) creates and returns `s2`, a string of the characters that repeat (occur more than once) in `s1`  

The decimal representation of binary 1111 is

??

The binary representation of decimal 24 is

??
Given the following Python code:

def q24(s):
    '''
    (??) -> ??
    Test function.
    >>> q24('The quick brown fox')
    ??
    >>> q24('Hello, world.')
    ??
    '''
    result = 999
    for i in range(len(s)):
        if s[i] == 'E' or s[i] == 'e':
            result = i
    return result
q24('Hello')

Complete the type contract for q24:
??

Executing this function will

a) Return the number of occurrences of 'e' in s, or 999 if none.
b) Return the number of occurrences of 'E' in s, or 999 if none.
c) Return the sum of a) and b), or 999 if none.
d) Return the position of the first occurrence of 'e' or 'E' in s, or 999 if none.
e) Return the position of the last occurrence of 'e' or 'E' in s, or 999 if none.

The first time the for loop executes, the value of i is

a) 'H'  b) 0  c) 1  d) 4  e) 5

The first time the for loop executes, the value of s[i] == 'E' or s[i] == 'e' is

a) 'E'  b) 'e'  c) True  d) False  e) 'False'

To determine this value, Python evaluated

a) s[i] == 'E'

b) s[i] == 'E'

c) s[i] == 'e'

d) result += 1
Given the following Python code:

```python
def q30(score):
    ''' exam function '''
    grade = 0
    if score >= 90:
        grade = 4
    elif score >= 80:
        grade = 3
    elif score >= 70:
        grade = 2
    elif score >= 60:
        grade = 1
    return grade
```

What is the result of executing `>>> q30(80)`?

a) 4  
 b) 3  
 c) 2  
 d) 1  
 e) NameError

Given the following UNTESTED Python code:

```python
def q3(myStr):
    '''final exam function'''
    newStr = ''
    for ch in myStr:
        if ch not in newStr:
            newStr += ch
    return newStr
```

What will be the result of executing `>>> q3('abab')`?

a) 'abab'  
b) 'ab'  
c) 'ba'  
d) 'a'  
e) 'b'
Given the following Python code:

```python
1  def isOdd(i):
2      '''(int) -> bool
3      exercise
4      '''
5      return i % 2 != 0
6  def q8(msg):
7      '''(str) -> ??
8      exercise
9      '''
10     odd_ct = 0
11     for ch in msg:
12         if isOdd(int(ch)):
13             odd_ct += 1
14     return odd_ct
15  def main():
16      '''exercise'''
17      code = '001100001100'
18      print(q8(code))
19      return None

Complete the type contract for q8:
??

What will be the result of executing
>>> main()

What would be the result of executing

print(msg) between lines 12 and 13?

a) NameError  b) 0  c) '001100001100'  d) str

print(ch) between lines 14 and 15 the first time the for loop is executed?

a) '0'  b) '1'  c) 'm'  d) 0  e) 1

print(i) between lines 5 and 6 the first time isOdd is executed?

a) '0'  b) '1'  c) 'm'  d) 0  e) 1
print(odd_ct) between lines 5 and 6 the first time isOdd is executed?

a) '0' b) '1' c) NameError d) 0 e) 1

print(code) between lines 24 and 25 when main is executed?

a) 0 b) 4 c) 8 d) NameError e) '0110001100'

print(msg) between lines 24 and 25 when main is executed?

a) 0 b) 4 c) 8 d) NameError e) '0110001100'

While function q8 is executing, odd_ct exists in a/the _____ namespace.

a) local b) global ('__main__') c) built-in

Functions main, q8, and isOdd exist in a/the _____ namespace.

a) local b) global ('__main__') c) built-in
When the following code is executed in the Python Shell,

```python
>>> mypi
??-1
>>> mypi = 3
>>> mypi
??-2
>>> w = mypi
>>> id(w) == id(mypi)
??-3
>>> type(w)
??-4
>>> mypi = [3]
>>> id(w) == id(mypi)
??-5
```

Replace ??-1 with the correct value:

a) 3  b) [1, 2, 3, 4]  c) True  d) False  e) NameError

Replace ??-2 with the correct value:

a) 3  b) [1, 2, 3, 4]  c) True  d) False  e) NameError

Replace ??-3 with the correct value:

a) 3  b) [1, 2, 3, 4]  c) True  d) False  e) NameError

Replace ??-4 with the correct value:

a) int  b) float  c) str  d) boolean  e) tuple

Replace ??-5 with the correct value:

a) 3  b) [1, 2, 3, 4]  c) True  d) False  e) NameError

Which statement is correct?

a) mypi is defined in the global namespace; w is defined in the global namespace.

b) mypi is defined in the global namespace; w is defined in the local namespace.

c) mypi is defined in the local namespace; w is defined in the global namespace.

d) mypi is defined in the local namespace; w is defined in the local namespace.

e) mypi is defined in the local namespace; w is no longer defined.
Given function `approx_sqrt`:

```python
def approx_sqrt(num, iterations):
    '''(number, int) -> float

Generates an approximate square root of num, a positive integer, via an iterative process that runs iterations times. The approximate square root is returned.

>>> approx_sqrt(1, 1)
1.0
>>> approx_sqrt(4, 1)
??-1
>>> approx_sqrt(4, 5)
2.000000000000002
'''
value = 1
for ctr in range(iterations):
    value = .5 * (value + num/value)
return ??-2
```

Replace `??-1` with the correct code:

a) 4  b) 1  c) 2  d) 2.0  e) 2.5

Replace `??-2` with the correct code:

a) num  b) value  c) 2  d) 2.0  e) 2.5

When `>>> approx_sqrt(4, 5)` is executed, the last value of `ctr` is

a) 0  b) 1  c) 4  d) 5  e) 2.000000000000002

`approx_sqrt` is an example of

a) recursion  b) Monte Carlo algorithm  c) accumulator pattern

d) encryption algorithm  e) REPL
Given the following UNTESTED Python code:

```python
def charCt(s, c):
    '''(str, str) -> int

    Return count of occurrences of char c in string s.
    >>> charCt('hello, world', 'o')
    2
    '''
    ct = 0
    for ch in s:
        if ch == c:
            ct += 1
    return ct
```

The set of test cases that will find the bug in `charCt` is

```plaintext
a) charCt('b','a')  b) charCt('abc','a')  c) charCt('abc', 'a')
charCt('a','a')  charCt('abc','b')  charCt('def', 'b')
charCt('abc','a')  charCt('abc','c')  charCt('ghi', 'x')
charCt('hi','o')  charCt('x', 'x')
```

After the following Python code is executed:

```python
>>> x = [99.9]
>>> y = x
>>> z = [99.9]
>>> y[0] = 0
```

The value of `x` is

```plaintext
a) 99.9  b) [99.9]  c) 0  d) [0]  e) None
```

The value of `y` is

```plaintext
a) 99.9  b) [99.9]  c) 0  d) [0]  e) None
```

The value of `z` is

```plaintext
a) 99.9  b) [99.9]  c) 0  d) [0]  e) None
```

The decimal representation of binary **11111** is:

```plaintext
a) 11,111  b) 63  c) 64  d) 31  e) 32
```
Given the following Python code:

```python
def taxable(inc, exempt, STD_E, STD_D):
    '''(number, int, number, number)
    Adjust gross income (inc) to taxable income
    by applying standard deduction and exemptions.
    CALLED BY: est_tax
    >>> taxable(20000, 1, 4150, 6500)
    9350
    '''
    #print(income)
    #print(salary)
    taxable_income = inc - STD_D
    exempt_adjust = STD_E * exempt
    taxable_income = taxable_income - exempt_adjust
    return taxable_income

def est_tax(income, exemptions):
    '''(number, int) -> None
    Generates an estimate for federal income tax.
    CALLS: taxable
    >>> est_tax(20000, 1)
    1870.0
    '''
    STD_EXEMPT = 4150
    STD_DEDUCT = 6500
    TAX_RATE = .20
    taxable_income = taxable(income, exemptions, STD_EXEMPT, STD_DEDUCT)
    estimated_tax = taxable_income * TAX_RATE
    #print('Estimated tax is:', estimated_tax)
    return None

def main(salary, exemptions):
    '''driver for estimated tax functions'''
    result = est_tax(salary, exemptions)
    print(result)
    print(salary)
    print(taxable_income)
    return None

salary = 20000
exemptions = 1
main(salary, exemptions)
```

1   def main(salary, exemptions):
2      '''driver for estimated tax functions'''
3      result = est_tax(salary, exemptions)
4      print(result)
5      print(salary)
6      print(taxable_income)
7      return None

salary = 20000
exemptions = 1
main(salary, exemptions)
After line 4 in main is executed, what will be printed?

a) 1870.0  b) 20000  c) None  d) NameError

After line 5 in main is executed, what will be printed?

a) 1870.0  b) 20000  c) None  d) NameError

After line 6 in main is executed, what will be printed?

a) 1870.0  b) 20000  c) None  d) NameError

If the #print(income) line of code in taxable were executed, what would be printed?

a) 1870.0  b) 20000  c) None  d) NameError

If the #print(salary) line of code in taxable were executed, what would be printed?

a) 1870.0  b) 20000  c) None  d) NameError

Given the following Python code:

```python
def q23(s):
    '''
    midterm function
    '''
    if len(s) == 1:
        return True
    elif len(s) == 2:
        return s[0] == s[1]
    elif s[0] != s[-1]:
        return False
    else:
        return q23(s[1:-1])

>>> q23('abcdeffedcba')
```

The second time q23 is called, the value of s is

a) True  b) False  c) 'abcdeffedcba'  d) 'bcdeffedcb'

The second time q23 is called, the value of s is

c) 'abcddcba'
Replace the ??s with the results of executing the following code in the Python Shell.

```python
>>> states = [2, 'OR', 'WA']
>>> states.append('ID')
>>> states[0] += 1
>>> states
??-1
>>> state = 'mt'
>>> state.upper()
??-2
>>> states.append(state)
>>> states[0] += 1
>>> states
??-3
>>> pnw = states.copy()
>>> states = states.append('CA')
>>> states[0] += 1
??-4
>>> pnw
??-5
```

Replace ??–1 with the correct result:

a) `[2, 'OR', 'WA']`  
   b) `[3, 'OR', 'WA', 'ID']`  
   c) `[3, 'ID', 'OR', 'WA']`

   d) `None`  
   e) `NameError`

Replace ??–2 with the correct result:

a) `'mt'`  
   b) `'MT'`  
   c) `None`  
   d) `TypeError`  
   e) `NameError`

Replace ??–3 with the correct result:

a) `[4, 'OR', 'WA', 'ID', 'mt']`  
   b) `[4, 'OR', 'WA', 'ID', 'MT']`

   c) `None`  
   d) `TypeError`  
   e) `NameError`

Replace ??–4 with the correct result:

a) `[4, 'OR', 'WA', 'ID', 'mt']`  
   b) `[4, 'OR', 'WA', 'ID', 'MT']`

   c) `None`  
   d) `TypeError`  
   e) `NameError`

Replace ??–5 with the correct result:

a) `[4, 'OR', 'WA', 'ID', 'mt']`  
   b) `[4, 'OR', 'WA', 'ID', 'MT']`

   c) `None`  
   d) `TypeError`  
   e) `NameError`
**TypeError, NameError, ZeroDivisionError** are examples of which type of error?

a) syntax       b) runtime       c) logical/semantic       d) regression       e) integrated

Given the following Python code:

```python
roman = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'M': 1000}
```  
What is the result of executing

```python
>>> roman['X']
```

a) True       b) False       c) 10       d) 'X'       e) TypeError

```python
>>> 'V' in roman
```

a) True       b) False       c) 5       d) 'X'       e) TypeError

```python
>>> 5 in roman
```

a) True       b) False       c) 'V'       d) 'X'       e) TypeError

Given the following Python code:

```python
1  def drawShape(s):
2      '''(int) --> None
3      Draw a square with sides of length s.
4      >>> drawShape(100)
5      [draws a square with sides length 100]
6      '''
7      turn = 90
8      for i in range(4):
9          fd(s)
10         lt(turn)
11      return None
```

Which lines of code would need to be changed to revise `drawShape` to draw an `n`-sided polygon, where `n` is a new argument to the function?

a) 1, 2, 14       b) 1, 2, 4, 6, 7, 9, 10       c) 1, 2, 4, 6, 7, 11, 12

```python
d) 1, 2, 4, 6, 7, 14       e) 1, 6, 14
```
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))
??

>>> q1(li)
??

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)
??

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]
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')
??
>>> q5('8')
??

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(tu, item):
    '''(tuple, object) ->

    ...'''
    for checkitem in tu:
        if checkitem == item:
            return True

    return False
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
    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:
        print('Average is', mean(li))
    else:
        print('Empty list')

    return None

>>> q6([1, 10, 4, 1])
??
>>> q6([])
??
```

For each of the two bugs,
-- circle the bug
-- indicate whether it is a syntax, runtime, logic, or documentation error
-- fix the bug
Write a function, `add_more_digits`, to sum the digits of a non-negative integer, \( n \). The sum is returned. All variables should be of type integer (no strings, lists, etc.).

The function should be written using CIS 210 style guidelines, except that the docstring examples of use may be omitted. Python code should clearly reflect the underlying algorithm. Code should use the most appropriate Python “tools” for solving the problem.

Practice (re-)writing code from any of the projects so far. Code should be very similar to posted solutions.