CIS 210 Fall 2020 Midterm Example Questions  KEY

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, class notes, labs, and readings from the text.

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

```
n = 5
mysum = 0
for ctr in range(1, n):
    mysum = mysum + ctr
print(mysum)
10
```

This code is an example of

a) accumulator pattern  
b) TypeError

c) conditional

d) indefinite iteration  
e) infinite loop

Given the following Python code:

```
# order of operations, type
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
from math import sqrt

def isInCircle(x, y, r):
    '''
    (x: number, y: number, z: 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: `bool`

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

a) ```python
d = sqrt(x**2 + y**2)
return d = r
```

b) ```python
d = sqrt(pow(x, 2) + pow(y, 2))
return d <= r
```

c) ```python
d = sqrt(x**2 + y**2)
return d < r
```

d) ```python
d = sqrt(pow(x, 2) + pow(y, 2))
isIn = d < r
return isIn
```

e) ```python
d = sqrt(x**2 + y**2)
return d
```

**Computational Problem Solving** - Replace ?? with the correct response.

**TASK/PROBLEM** ➔ **Computational Thinking** ➔ **SOLUTION/ALGORITHM/COMPUTATIONAL PROCESS**

**ALGORITHM** ➔ **Design/Coding** ➔ **COMPUTER PROGRAM**

**COMPUTER PROGRAM** ➔ ?? ➔ **HIGH QUALITY (RELIABLE/REUSABLE) COMPUTER PROGRAM**

a) looping b) executing c) printing d) testing and debugging
Given the following Python code:

def q29(s1):
    '''(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

Given:

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

def f(x, y):
    '''(x: int, y: 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 q1(astr):
    '''(astr: str) -> ??'''
    slen = len(astr)
    ctr = 0
    for ch in astr:
        ctr += 1
    return slen == ctr

astr = 'CIS 2xx CIS 3xx'

>>> type(q1(astr))
bool

>>> q1(astr)
True

for is a Python

  a) keyword   b) identifier   c) literal object   d) library module

q1 is a Python

  a) keyword   b) identifier   c) literal object   d) library module

ctr is a Python

  a) keyword   b) identifier   c) literal object   d) library module

0 is a Python

  a) keyword   b) identifier   c) literal object   d) library module

return is a Python

  a) keyword   b) identifier   c) literal object   d) library module

'CIS 2xx CIS 3xx' is a Python

  a) keyword   b) identifier   c) literal object   d) library module
Given the following Python code:

```python
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:

```python
(s: str) -> int
```

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 '''
    gradepoint = 0
    if score >= 90:
        gradepoint = 4
    if score >= 80:
        gradepoint = 3
    if score >= 70:
        gradepoint = 2
    if score >= 60:
        gradepoint = 1
    return gradepoint
```

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: # executing functions; parameter passing

```python
1 def isOdd(i):
2     '''(i: int) -> bool
3     exercise
4     return i % 2 != 0
5
6 def q8(msg):
7     '''(msg: str) -> ??
8     exercise
9     odd_ct = 0
10    for ch in msg:
11       if isOdd(int(ch)):
12          odd_ct += 1
13    return odd_ct
14
15 def main():
16     '''exercise'''
17     code = '001100001100'
18     print(q8(code))
19     return
```

Complete the type contract for q8:
```
int
```

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

```
4
```

What would be the result of executing
```python
print(msg) between lines 12 and 13?
```

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

```python
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

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

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

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

- a) 0
- b) 4
- c) 8
- d) NameError
- e) '001100001100'
**Basic code tracing - what happens when a function is executed. Accumulator pattern.**

Given function `approx_sqrt`:

```python
def approx_sqrt(num, iterations):
    '''(num: number, iterations: 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
Debugging; test cases that generate different results are needed.

def charCt(s, c):
    '''(s: str, c: 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 NOT find the bug in charCt is

a) charCt('', 'a')
   charCt('abc', 'a')
   charCt('hi', 'o')
   charCt('hello, world', 'o')

b) charCt('a', 'a')
   charCt('abc', 'b')
   charCt('ghi', 'x')
   charCt('x', 'x')

c) charCt('abc', 'a')
   charCt('def', 'b')
   charCt('ghi', 'x')
   charCt('x', 'x')

Python conditional statements
Given the following Python code:

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

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

a) 4    b) 3    c) 2    d) 1    e) NameError
Given the following UNTESTED Python code:  

Be aware of what functions return.

def taxable(inc, exempt, STD_E, STD_D):
    '''(inc: number, exempt: int, STD_E: number, STD_D: 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):
    '''(income: number, exemptions: 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, exemption, STD_EXEMPT, STD_DEDUCT)
    estimated_tax = taxable_income * TAX_RATE
    return estimated_tax

1   def main(salary, exemptions):
2      '''driver for estimated tax functions'''
3      result = est_tax(salary, exemptions)
4      print(result)
5      print(salary)
6      return

salary = 20000
exemptions = 1
main(salary, exemptions)
When function `taxable` is called from `est_tax`, which of the following happens:

a) an activation record for function `taxable` is added to the call stack

   **Yes**  **No**

a) the activation record for function `est_tax` is removed from the call stack

   **Yes**  **No**

b) `inc` and `exempt` are assigned values

   **Yes**  **No**

c) NameError for incorrect names STD_D and STD_E

   **Yes**  **No**

d) TypeError as `taxable` is not an executable type

   **Yes**  **No**

e) the taxable income is printed

   **Yes**  **No**

**Types of errors.**

`TypeError`, `NameError`, `ZeroDivisionError` are examples of which type of error?

a) syntax  b) **runtime**  c) logical/semantic  d) regression  e) integrated
Revising code.

Given the following Python code:

```python
1  def drawShape(s):
2      '''(s: 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
```

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   d) 1, 2, 4, 6, 7, 14   e) 1, 6, 14
Given the following Python code:

```python
from math import pi

def circle_area(rad):
    """(rad: num) -> float
    return the area of a circle with radius rad
    >>> circle_area(1)
    3.1
    >>> circle_area(2)
    12.6
    ""
    area = pi * rad ** 2
    area = round(area, 1)
    return area

def pizza_calculator(diameter, cost):
    """(diameter: int, cost: num) -> float
    Calculates and returns the cost per square inch of pizza for a pizza of given diameter and cost.
    >>> pizza_calculator(14, 18)
    0.117
    >>> pizza_calculator(14, 20.25)
    0.132
    ""
    r = diameter / 2
    area = pi * r ** 2
    cost_per_inch = cost / area
    cost_per_inch = round(cost_per_inch, 3)
    return cost_per_inch

Which line of code below should replace lines 13 and 14 in `pizza_calculator` so that the pizza area calculation is moved to `circle_area`?

a) `area = circle_area(diameter / 2)`
b) `area = circle_area(rad)`
c) `circle_area(diameter / 2)`
d) `circle_area(rad)`

Draw a picture of the Python call stack / activation records at line 12 when `pizza_calculator(14, 18)` is executed:

**global frame:**
- `circle_area ➔ function`
- `pizza_calculator ➔ function`

**pizza_calculator:**
- `diameter ➔ 14`
- `cost ➔ 18`
The Taylor series expansion for $e^x$ has the following form:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

If we truncate the sum after $N$ terms, we have an approximation for $e^x$ as:

$$e^x \approx \sum_{n=0}^{N-1} \frac{x^n}{n!}$$

Given the following Python code, consistent with this approximation for $e^x$:

```python
from math import factorial  # factorial(n) returns n!

def approx_e(x, i):
    '''(x: number, i: int) -> ??-1

    Generate approximation for e**x with i iterations of the Taylor
    series expansion for e**x. The approximate value of e**x is returned.
    
    >>> approx_e(1, 100)
    2.7182818284590455
    '''
    acc = 0
    for n in range(??-2):
        acc += pow(x, n)/factorial(n)
    return acc

Complete the docstring (??-1):

a) float  b) int  c) number (int or float)  d) bool  e) str

Complete the call to Python `range` (??-2):

a) i  b) i - 1  c) i + 1  d) x  e) x - 1

When `>>> def = 123` and `>>> abs = 456` are executed, the result is

a) new value for def / new value for abs  b) new value for def / error

c) error / new value for abs  d) error / error

Python functions __??__ return a value and __??__ cause a side effect.

a) always/never  b) always/sometimes  c) always/always
d) sometimes/always  e) sometimes/sometimes
What is the result of executing the following Python code:

def twice(x):
    '''midterm function'''
    result = 2 * x
    #print(result)
    return

def thrice(x):
    '''
    result = twice(x) + x
    return result

>>> thrice(5)
??

a) 5     b) 10     c) 15     d) TypeError     e) NameError
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).

The function should be written using CIS 210 style guidelines; the brief description (only) may be omitted from the docstring (include type contract and at least one example of use). Python code should clearly reflect the underlying algorithm. Code should use only the most appropriate Python "tools" for solving the problem.

```python
def add_more_digits(n):
    '''(n: int) -> int

    >>> add_more_digits(12345)
    15
    '''
    digit_sum = 0
    while n > 0:
        digit = n % 10
        n = n // 10
        digit_sum += digit
    return digit_sum
```

Write function `poly`, with three parameters, `num_sides`, `side_len`, and `pcolor`, which will draw a polygon with `num_sides` sides of length `side_len`, using fill color `pcolor` if one is provided, and no fill color otherwise (i.e., `pcolor` has a default value of the empty string: '').

Write function `house` that calls `poly` to draw a house with a square first floor and triangle roof.

A parcel is thrown downward at a speed of `v` meters per second from an airplane at altitude 11000 m. As it falls, its distance from the ground is given by the formula 
\[
-4.9t^2 - vt + 11000,
\]
where `t` is the time in seconds since it was dropped. Write a function, `gravity`, to print the time it will take to reach the ground, rounded to two decimal places. (P2b – Python Variations, from CS Circles)

Function `gravity` should have one parameter, `v`, a positive floating point (decimal) number

The required time is given by the quadratic formula

\[
\frac{-v + \sqrt{v^2 - 4(-4.9)(11000)}}{2(-4.9)}
\]