1-2. Given the Python statement

```python
greeting = 'Hello, CIS 210.'
```

1. [1 pt.] `greeting` is a Python
   a. variable  b. algorithm  c. operator  d. function

2. [1 pt.] 'Hello, CIS 210.' is a(n)
   a. expression  b. string literal  c. neither of these  d. both of these

3. [1 pt.] To use the `sqrt()` function, you must first use the following Python statement:
   a. `use math`  b. `import math`  c. `math::sqrt()`  d. `return sqrt()`

4. [2 pts.] Given the following Python code:

   ```python
   1. >>> name = 'Roy G Biv'
   2. >>> i = name[1]
   3. >>> i
   4. >>> m = name[4]
   5. >>> m
   6. >>> name[-1]
   ```

   What is printed after lines 3, 5, and 6?
   a. 'R', 'G', 'v'
   b. 'R', ', ', 'v'
   c. 'o', 'G', 'i'
   d. 'o', 'G', 'v'

5 - 6. Given the following Python code:

```python
for i in range(10):
    print('check')
```

5. [1 pt.] How many times will `check` be printed?
   a. 0  b. 1  c. 9  d. 10

6. [1 pt.] What will be the value of `i` the last time the loop is executed?
   a. 0  b. 1  c. 9  d. 10
7. [2 pts.] We've heard from the client who requested the federal income tax estimator. Recall that the estimated tax is calculated as taxable income (gross income minus the sum of the standard deduction and the exemption amount * number of exemptions) multiplied by the tax rate:

\[
\text{exemption} = 4050 \\
\text{std\_deduction} = 10000 \\
\text{tax\_rate} = .20 \\
\text{taxable\_income} = \text{gross\_income} - \text{std\_deduction} \\
\text{exempt\_adjust} = \text{exemption} \times \text{number\_exempt} \\
\text{taxable\_income} = \text{taxable\_income} - \text{exempt\_adjust} \\
\text{estimated\_tax} = \text{taxable\_income} \times \text{tax\_rate}
\]

They forgot to mention that if the gross income is less than the sum of the standard deduction and exemption adjustment, the function should return an estimated tax of zero (not a negative number, as it currently does).

To implement this new functionality, we will need to use

a. math module  
   b. random module  
   c. conditional statement  
   d. for loop

8-10. Given the following Python code:

```python
def mysqrt(n, k):
    '''(integer, integer) -> ??-8
    Generates k successive, better approximate square roots of n, a positive integer.
    The approximate square root is ??-9.
    >>> mysqrt(25, 5)
    5.000023178253949
    '''
    # A(n) ??-10 process calculates approximate square root.
    approx_val = 1
    for ctr in range(k):
        approx_val = .5 * (approx_val + n/approx_val)
    return approx_val
```

8. [1 pt.]
a. integer  
   b. float  
   c. string  
   d. None

9. [1 pt.]
a. returned  
   b. printed  
   c. returned and printed  
   d. neither returned nor printed

10. [1 pt.]
a. iterative  
    b. Monte Carlo  
    c. cipher  
    d. infinite loop
11-12. Given the following Python code:

```python
def qx1(a):
    ''' Midterm 1
    ...
    a += 1
    return a

def qx2(a):
    a += 1
    print(a)
    return 100

def qx3(x):
    result = qx1(x) + qx2(x)
    return result
```

11. [2 pts.] Which is correct?

<table>
<thead>
<tr>
<th></th>
<th>a.</th>
<th>b.</th>
<th>c.</th>
<th>d.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>a = 5</td>
<td>a = 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td>qx1(a)</td>
<td>qx1(a)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c.</td>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d.</td>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12. [2 pts.] Which is correct?

<table>
<thead>
<tr>
<th></th>
<th>a.</th>
<th>b.</th>
<th>c.</th>
<th>d.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>qx3(0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td>qx3(0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c.</td>
<td>qx3(0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d.</td>
<td>qx3(0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13. [2 pts.] Given the following Python code:

```python
def q13(s1):
    ''' midterm func
    ...
    s2 = ''
    for ch in s1:
        if ch not in s2:
            s2 += ch
    return s2
```

Which is correct?

<table>
<thead>
<tr>
<th></th>
<th>a.</th>
<th>b.</th>
<th>c.</th>
<th>d.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>q13('123456')</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td>q13('123456')</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c.</td>
<td>q13('123456')</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d.</td>
<td>q13('123456')</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>'123456'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>'1'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>'1'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>'1'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>q13('999999')</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>q13('999999')</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>q13('999999')</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>q13('999999')</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>'999999'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>'9'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>'9'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>'9'</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
14. [2 pts.] Given the following Python code:

```python
import math
import random

def isInCircle(x, y, r):
    '''(number, number, number) -> Boolean

    Returns True if point (x, y) is in
    the circle centered at (0,0) with radius r.
    >>> isInCircle(0, 0, 1)
    True
    >>> isInCircle(1, 2, 1)
    False
    '''
    d = math.sqrt(x**2 + y**2)
    isIn = d <= r
    return isIn

def montePi(numDarts):
    '''
    (integer) -> float
    Uses a Monte Carlo algorithm (looping
    numDarts times) to generate an
    approximate value for pi, which is returned.
    For example,
    >>> montePi(100000)
    3.13572
    '''
    inCircle = 0
    for i in range(numDarts):
        x = random.random()
        y = random.random()
        d = math.sqrt(x ** 2 + y ** 2)
        if d <= 1:
            inCircle += 1
    approxPi = inCircle/numDarts * 4
    return approxPi
```

Which lines of function `montePi` would need to be changed to use function `isInCircle`?

a. 19, 21  
b. 3, 19, 21, 22  
c. 13, 22  
d. 3, 10, 13, 22
15. [10 pts. – per usual rubric] Write a Python function `charCount`, which will partially implement the string method `count` — without using the string method `count`. `charCount` will have two parameters, `s`, a string, and `c`, a character (string of length 1), and return the number of occurrences of `c` in `s`.

Your function should be written using CIS 210 style guidelines, including a complete docstring (type contract, brief description including referencing parameters by name, mention of returned value and side effects if any, and examples of function use. It should work for your own simple examples (include at least one) as well as the following test cases:

```
>>> charCount('hello, world', 'o')
2
>>> charCount('hello, world', 'z')
0
```

def charCount(s, c):
    '''
    (string, string) -> integer

    Returns number of occurrences of c (a character) in s.
    >>> charCount('eeeee', 'e')
    5
    >>> charCount('', 'e')
    0
    >>> charCount('eeeee', 'a')
    0
    >>> charCount('abcba', 'a')
    2
    >>> charCount('hello, world', 'o')
    2
    >>> charCount('hello, world', 'z')
    0
    '''
    ct = 0
    for ch in s:
        if ch == c:
            ct += 1
    return ct