(Q2) Given the following Python code:

```python
def fb(n):
    '''(n: int) -> None
    Play fizzbuzz up to n, a positive integer. Results are printed during play.
    >>> fb(15)
    1
    2
    fizz
    4
    buzz
    fizz
    ...
    fizzbuzz
    '''
    for i in range(1, n+1):
        m3 = (i % 3) == 0
        m5 = (i % 5) == 0

        line1:  # missing line
            print('fizzbuzz')
        line2:  # missing line
            print('fizz')
        line3:  # missing line
            print('buzz')
        else:
            print(i)
    print('Game over!')
    return
```

Choose the missing lines of code from the following:

(a) `elif m3`
(b) `elif m5`
(c) `if (i % 3) and (i % 5) == 1`
(d) `if m3 and m5`
(e) `if m3 or m5`
(f) `if m3 == True or m5 == True`
(g) `if m3 or m5 == True`

line1:

line2:

line3:
(Q3) Match the code with the result of executing the code (results may be used more than once)

(A)
```python
li1 = [99, 100, 101]
li2 = li1
li3 = li1.copy()
li1[1] = 'hi'
print(li1)
print(li2)
print(li3)
```

(1)
```python
[99, 100, 101]
['hi', 100, 101]
[99, 'hi', 101]
```

(B)
```python
li1 = [99, 100, 101]
li2 = li1
li3 = li1.copy()
li2.append('hi')
print(li1)
print(li2)
print(li3)
```

(2)
```python
[99, 100, 101]
[99, 'hi', 101]
[99, 'hi', 101]
```

(C)
```python
li1 = [99, 100, 101]
li2 = li1
li3 = li1.copy()
li1 = li1.append('hi')
print(li1)
print(li2)
print(li3)
```

(3)
```python
[99, 100, 101]
None
[99, 100, 101, 'hi']
```

(4)
```python
[99, 100, 101, 'hi']
[99, 100, 101, 'hi']
[99, 100, 101]
```

(Q4) The results in Question 3 can be explained by (choose all that apply):

(a) lists are a mutable data type
(b) lists are an immutable data type
(c) strings are a mutable data type
(d) strings are an immutable data type
(e) reference semantics
(f) many list methods update the list as a side effect and return None
(Q5) Given the following Python function header, docstring, and first line of code:

def myMode(alist):
    '''(alist: list of numbers) -> list

    Return mode(s) of alist. like statistics module multimode.

    Calls: genFreqTable (which you may assume is available: given a list, generates a dictionary with list items as keys and the number of occurrences of the item in the list as the value of each key)

    >>> myMode ([1, 2, 2, 3, 99])      #normal
    [2]
    >>> myMode ([99])                  #border - list len 1
    [99]
    >>> myMode ([0, 0, 1, 1])          #all values are a mode
    [0, 1]
    >>> myMode ([5, 7, 1, 3])          #all values are a mode
    [5, 7, 1, 3]
    ...
    modeli = []

Order the following lines of code to finish implementing myMode.

1 countd = genFrequencyTable(alist)
2 maxct = max(countli)
3 return modeli
4     modeli.append(k)
5 for k in countd:
6     if countd[k] == maxct:
7     countli = countd.values()

(Q6) Given the following UNTESTED Python code:

def mymin(li):
    '''(li: list) -> int

    Return least integer from a non-empty list of integers.
    '''
    current_min = li[0]
    for item in li[1:]:
        if item < current_min:
            current_min = item
    return current_min

Which of the following calls to execute mymin will detect the bug(s) (choose all that apply)?

>>> mymin([4, 5, -1, 6])
>>> mymin([1, 2, 3, 4])
>>> mymin([-99])
>>> mymin([-1, -2, -3])
>>> mymin([0, 0, 0])
(Q7) For the test case(s) that will detect the bug(s), give the result(s) they will return (separated by a single space between each result if there is more than one result):

??

(Q8) This is an example of which type of error?

a) syntax   b) semantic/logical   c) runtime   d) documentation

(Q9) When the following code is executed in the Python Shell,

```python
>>> def auxf(s):
    
```

"auxiliary function"
```
    s = s * 2
    return s
```

1 >>> t = 123
2 >>> t = auxf(t)
3 >>> t = '123'
4 >>> t = auxf(t)
5 >>> print(t)
```

Which statement is correct when s = s * 2 in auxf is executed the first time (auxf called in line 2)?

a) t is defined in the global namespace; s is defined in the global namespace.

b) t is defined in the global namespace; s is defined in the local namespace.

c) t is defined in the local namespace; s is defined in the global namespace.

d) t is defined in the local namespace; s is defined in the local namespace.

e) t is defined in the global namespace; s is no longer defined.

Which statement is correct when print(t) is executed (line 5)?

a) t is defined in the global namespace; s is defined in the global namespace.

b) t is defined in the global namespace; s is defined in the local namespace.

c) t is defined in the local namespace; s is defined in the global namespace.

d) t is defined in the local namespace; s is defined in the local namespace.

e) t is defined in the global namespace; s is no longer defined.
The `<value>` / `<type>` of `t` after the first call to `auxf(t)` will be `<??>/<??>` the value/type of `t` after the second call to `auxf(t)`?

a) same as / same as

b) different from / different from

c) same as / different from

d) different from / same as

This demonstrates which characteristic(s) of Python?

a) strong typing   b) weak typing   c) dynamic typing
d) static typing   e) conditionals

(Q10) Given the following Python code:

```python
def q10a(i):
    '''(int) -> ??
    final exam function
    '''
    return i % 2 != 0

def q10b(i):
    '''(int) -> ??
    final exam function
    '''
    return i % 2 == 0

def q10(f, msg):
    '''(function, list) -> None
    final exam function
    '''
    for i in range(len(msg)):
        if f(i):
            msg[i] = msg[i].upper()
    return

def main():
    '''final exam function'''
    li = ['hello', 'world', 'hello', 'CIS210']
    q10(q10a, li)
    print(li)
    q10(q10b, li)
    print(li)
    return
```

1  def main():
2     '''final exam function'''
3     li = ['hello', 'world', 'hello', 'CIS210']
4     q10(q10a, li)
5     print(li)
6     q10(q10b, li)
7     print(li)
8     return
(Q11) Given the following Python code:

def isMemberI(seq, n):
    '''(seq: sequence, n: item) -> bool

    Use binary search to check for membership
    of int n in sorted sequence seq. Return True
    if n is a member, else return False.

    Note: seq is not maintained in its original
    form inside this function.

    >>> isMemberI((1, 2, 3, 3, 4), 4)
    True
    >>> isMemberI([1, 2, 3, 3, 4], 2)
    True
    >>> isMemberI('aeiou', 'i')
    True
    >>> isMemberI('aeiou', 'y')
    False
    '''
    if len(seq) == 0:
        return False

    while len(seq) > 0:
        mid = len(seq) // 2

        if seq[mid] == n:
            return True
        elif seq[mid] > n:
            seq = seq[:mid]
        else:
            seq = seq[mid+1:]

    return False

For a sequence seq where len(seq) == 1000 and n is NOT an item in seq, the while loop would execute approximately how many times?

a) 0  b) 1  c) 10  d) 100  e) 1000
Given the following UNTESTED Python code:

```
CIS 210 Fall 2020 FINAL EXAM
Add a parity bit to binary representation of characters.

def parity(bitrep):
    '''(bitrep: str) -> str
    Determine (even) parity for bit representation (bitrep); return result (0 or 1).
    >>> parity('1100011')
    '0'
    >>> parity('1100100')
    '1'
    >>> parity('0000000')
    '0'
    >>> parity('1111111')
    '1'
    '''
    # parity bit starts at 0
    p = 0
    for bit in bitrep:
        if bit == '1':
            p += 1
    if p % 2 == 0:
        p = '0'
    else:
        p = '1'
    print(p)
    return

def encode(letter):
    '''(letter: str) -> str
    add a parity bit to letter (most significant digit); return encoded letter.
    calls: parity, to get the parity bit
    >>> encode('c')
    '01100011'
    >>> encode('d')
    '11100100'
    '''
    ch_bits = bin(ord(letter))
    ch_bits = ch_bits[2:]
    parity_bit = parity(ch_bits)
    encoded_letter = parity_bit + ch_bits
    return encoded_letter
```

(Q12)
def main():
    letter = 'c'
    print(f'Parity bit for {letter} is {encode(letter)[0]}.')
    return

main()

What would be result of executing this program (Run Module)? (Choose all that apply.)

a) functions main, encode, and parity would be defined in the global frame

b) function main would be defined in the global frame; functions parity and encode would be defined in a local namespace.

c) the program would execute and Python would print the result: Parity bit for c is 0.

d) the program would execute and Python would print the result: Parity bit for c is 1.

e) the program would execute and Python would report a runtime error: TypeError

f) the program would execute and Python would report a runtime error: NameError

g) the program would execute and Python would print a value: 0

h) the program would execute and Python would print a value: 1

(Q13) Given the following Python code:

```python
1 def foo(x):
2     x.pop()
3     return

4 def bar(x):
5     foo(x)
6     y = foo(x); print(x, y)
7     x = foo(x)
8     return
```

```python
>>> x = ['CIS 210', 'CIS 211', 'CIS 212']
>>> bar(x)
>>> print(x)
```

(a) When `bar(x)` is called, what is printed for the value of `x` when line 6 is executed??

(b) When `bar(x)` is called, what is printed for the value of `y` when line 6 is executed??

(c) What is printed when `print(x)` is called after `bar(x)` has finished executing??

(Q14) `>>> print(y)` would cause an error. This error is due to

a) static typing       b) dynamic typing       c) variable scope       d) mutable data types
(Q15) This error would be an example of what kind of error?

a) syntax   b) semantic/logical   c) runtime   d) documentation

(Q16) If functions `foo` and `bar` (from Q13) are updated:

```python
1  def foo(x):
2      y = x.pop()
3      return y

4  def bar(x):
5      foo(x); print(x)
6      y = foo(x); print(x, y)
7      x = foo(x); print(x)
8      return
```

```python
>>> x = ['CIS 210', 'CIS 211', 'CIS 212']
>>> bar(x)
>>> print(x)
```

When `bar(x)` is called

(a) what is printed for the value of `x` when line 5 is executed??

(b) what is printed for the value of `y` when line 6 is executed??

(c) what is printed for the value of `x` when line 7 is executed??

(Q17) Briefly explain the new result (1-3 sentences): ??

(Q18) Given the following Python code:

```python
1  def foo(x):
2      x = x.lower()
3      return x

4  def bar(x):
5      foo(x)
6      y = foo(x); print(x, y)
7      x = foo(x) 
8      return 
```

```python
>>> x = 'CIS 210'
>>> bar(x)
>>> print(x)
```

(a) When `bar(x)` is called, what is printed for the value of `x` when line 6 is executed??

(b) When `bar(x)` is called, what is printed for the value of `y` when line 6 is executed??

(c) What is printed when `print(x)` is called after `bar(x)` has finished executing??

(Q19) Briefly explain the results (1-3 sentences): ??
(Q20) Write a program to report the daily rainfall for the first 15 days of the month.

Write a function, `rainData`, with one parameter, `f`, the name of a file containing rain data (daily rainfall in Eugene from Month Day 1 through Month Day 15).

The file, rainfall.txt, has two lines: a header line, and another line of space-separated values, which are the recorded daily rainfall amounts for the first 15 days of the month (the first day had .2 inches of rain, the second day had 4 inches of rain, and so on):

```
#rainfall data for final exam CIS 210 F20 mm/01-15/yyyy Eugene, OR
.2 4 .16 .2 .16 .15 .2 .12 .15 .16 .16 .16 .15 .16 .2
```

Function `rainData` should create and return a rain dictionary, where the dictionary keys are the days of the month (1 through 15) and the associated value is the rainfall for that day.

Next write a function, `reportOneDay`, with two parameters, `date` and `rdict`. `date` is an integer between 1 and 15 inclusive and `rdict` is the dictionary created by `rainData`. `reportOneDay` should report (print) the rainfall for date, for example:

The rainfall on day 3 was 0.16 inches.

If the rainfall for the date is greater than or equal to 4 inches, `reportOneDay` should also report:

HEAVY RAIN ALERT

`reportOneDay` should return `None`.

Finally, write a main function, which will call `rainData` to create the rain data dictionary, and then call `reportOneDay` to report the rainfall for each day for the first 15 days of the month. `main` will have no parameters and should return `None`.

Code should be written according to CIS 210 style guidelines, including a docstring with a type contract, brief description, and at least one example of use. (The docstring may be omitted in the main function.) Examples of use in `rainData` and `reportOneDay` may refer to a small example file, for example,

```
#rainfall_short.txt
.2 4
```

(Q21) What is the result when the following Python code is executed?

```python
>>> adj1 = 'best'
>>> adj2 = 'pleasant'
>>> adj3 = 'winter'
>>> print(f'{adj1.capitalize()} wishes for a {adj2} {adj3} break. ')
```

a) Best wishes for a pleasant winter break.
b) best wishes for a pleasant winter break.
c) {adj1.capitalize()} wishes for a {adj2} {adj3} break.
d) BEST WISHES FOR A PLEASANT WINTER BREAK.