(0) (3 pt.) What are the values of a, b, and c after the following code is entered into the Python shell?

```python
>>> a = 10
>>> b = a / 3
>>> c = round(b)
>>> a = a ** 2
```

```plaintext
>>> a
100
>>> b
3.33333...
>>> c
3
```

(1) (4 pts.) Given the following Python code:

```python
def mystery(s1, s2):
    """what does this function do?""
    print(s1, s2)
    result1 = len(s1)
    result2 = len(s2)
    final_result = result1 + result2
    print(s1, s2)
    return final_result
```

```plaintext
>>> mystery('Pacific,', 'Klamath')
Pacific, Klamath
Pacific, Klamath
15
```

(a) What will be displayed in the Python Shell?
`Pacific, Klamath
Pacific, Klamath
15`

(b) Write a type contract for function mystery:

```plaintext
(sequence) -> int or (list) -> int or (str) -> int
```
(2) (2 pts.) Given:

\[ x = 20 \\
\text{y} = 10 \ // \ 5 \\
z = 10 \ % \ 2 \\
\]

What will be the result of executing the following code?

```python
>>> (x < y) and (x < z)
False
```

All the parts of question (3) are a group and should be answered in order:

(3) (8 pts.) After the following code is entered into the Python Shell:

```python
>>> ctemp = 90
```

what will be the result of executing the following code?

```python
>>> ctemp
90

>>> 'ctemp' * 2
'ctempctemp'

>>> to_boil = 100 - ctemp
[to_boil gets the value 10, but Python does not display anything]

>>> to_boil <= ctemp
True

>>> to_boil ** 2
100

>>> ctemp // 9
10

>>> [ctemp, to_boil, str(ctemp)]
[90, 10, '90']
```

(4) (2 pts.) What will be the output after the following Python code is executed:

```python
>>> classinfo = [['name', 'CIS122'], ['CRN', '31461'], ['room', 'PAC123']]

>>> for info in classinfo:
    if info[0] == 'CRN':
        print(info)
['CRN', '31461']
```
All the parts of question (5) are a group and should be answered in order:

(5) (8 pts.) When a user enters

```python
>>> states = ['Oregon', 'Washington', 'California', 'Idaho', 'Montana']
>>> states
```

into the Python shell, what does Python return?

a) ['Oregon', 'Washington', 'California', 'Idaho', 'Montana']

b) states
c) 'states'
d) None

If a user then enters

```python
>>> for item in states: 
    print(item[1])
```

into the Python shell, what does Python print?

a) ['Oregon', 'Washington', 'California', 'Idaho', 'Montana']

b) states
c) 'states'
d) None

If a user then enters

```python
>>> example = states.reverse()
>>> example
```

into the Python shell, what does Python return?

a) ['Oregon', 'Washington', 'California', 'Idaho', 'Montana']

b) ['Montana', 'Idaho', 'California', 'Washington', 'Oregon']

c) ['nogerO', 'notgnihsaW', 'ainrofilaC', 'ohadI', 'anatnoM']

d) None

If a user then enters

```python
>>> states
```

into the Python shell, what does Python return?

a) ['Oregon', 'Washington', 'California', 'Idaho', 'Montana']

b) ['Montana', 'Idaho', 'California', 'Washington', 'Oregon']

c) ['nogerO', 'notgnihsaW', 'ainrofilaC', 'ohadI', 'anatnoM']

d) None
6) (3 pts.) Given this data in a file called **planet_data.txt**:

Mercury Mars
Neptune
Venus

What is the output after the following Python code is executed?

```python
>>> with open('planet_data.txt', 'r') as planetf:
    planet = planetf.read()

>>> planet
'Mercury Mars\nNeptune\nVenus'
```

(7) (1 pts.) A difficult programming problem is one that is

a) precisely specified     b) **ambiguous**     c) uses turtle module     d) none of these

(8) (3 pts.) Given the following Python code:

```python
from turtle import *

def drawpoly(s, len):
    '''Draws a polygon with s sides of length len.''

    angle = 360.0/s
    ctr = 0

    while ctr < s:
        for ctr in range(s):
            fd(len)
            rt(angle)
            ctr = ctr + 1
```

Mark the code with changes needed to replace the while loop with a for loop.

(9) (4 pts.) Given:

```python
>>> outer = ['Li', 'Na']
>>> inner = ['F', 'Cl']

>>> for metal in outer:
    for halogen in inner:
        print(metal + halogen)
```

What will be the result of executing the above code?

**LiF**
**LiCl**
**NaF**
**NaCl**
Given the following Python function:

```python
def boggleIdeas(astring):
    """
    return True if astring
    -- is all alphabetic characters
    -- is longer than 2 characters
    -- does not appear in a list of frequently used words:
       ['cat', 'hat', 'sat', 'bee', 'see', 'tee', 'bet', 'let']
    """
    if not astring.isalpha():
        return False
    if len(astring) <= 3:
        return False
    if astring in ['cat', 'hat', 'sat', 'bee', 'see', 'tee', 'bet', 'let']:
        return False
    return True
```

Identify the following:

(a) function name
   **boggleIdeas**

(b) function parameter(s)
   **astring**

(c) Write the type contract for boggleIdeas:
   **(str) -> boolean**

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

(d) print(boggleIdeas('cat'))
   **False**

e) print(boggleIdeas('test') == 'False')
   **False**

(f) Give another good test case for boggleIdeas (i.e., one that reveals the bug in the code):
   **boggleIdeas('jet'), for example (len = 3 and all alpha and not in freq list)**

(g) Fix the bug in function boggleIdeas (precise words or mark the code):
   ```python
   if len(astring) < 3:
   ```
(11) (4 pts.) Given the following Python function:

def anyupper(t):
    """ What does this code do? """
    res = []
    for item in t:
        if item.isupper():
            res.append(item)
    return res

mystring = 'Practical Programming'

What will be the output after the following Python code is executed:

```python
>>> anyupper(mystring)
['P', 'P']
```

```python
>>> mystring
'Practical Programming'
```

(12) (4 pts.) Given the following Python function:

def trackscore(score):
    """ convert score to grade point; return grade point """
    gradepoint = 0
    if score >= 90:
        gradepoint = 4
    elif score >= 80:
        gradepoint = 3
    elif score >= 70:
        gradepoint = 2
    elif score >= 60:
        gradepoint = 1
    return gradepoint

(a) Show the result of executing the following Python code:

```python
>>> gp = trackscore(90)
>>> gp
1
```

(b) Tell how this bug can be fixed (precise words or code):

*elifs instead of ifs (after first if)*
(13) (2 pts.) Show the result of executing the following Python code:

```python
seasons = ['winter', 'spring', 'summer', 'fall']
for i in range(len(seasons)):
    seasons[i] = seasons[i].capitalize()
print(seasons)
['Winter', 'Spring', 'Summer', 'Fall']
```

(14) (2 pts.) Given the following Python code:

```python
def two_numbers(astring):
    # (str) -> Boolean
    return True if astring has at least two numbers, otherwise return False

>>> two_numbers('CIS122')
True
>>> two_numbers('Ducks')
False
>>> two_numbers('ABC-1')
False
```

digits_ctr = 0
for c in astring:
    if c.isdigit():
        digits_ctr += 1

    return (digits_ctr >= 2)

Circle the column with the better set of test cases for function `two_numbers`:

<table>
<thead>
<tr>
<th><code>two_numbers('12')</code></th>
<th><code>two_numbers('he99llo')</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>two_numbers('9')</code></td>
<td><code>two_numbers('A12B')</code></td>
</tr>
<tr>
<td><code>two_numbers('AB3')</code></td>
<td><code>two_numbers('goodbye')</code></td>
</tr>
<tr>
<td><code>two_numbers('</code>)`</td>
<td><code>two_numbers('C34D')</code></td>
</tr>
<tr>
<td><code>two_numbers('$444')</code></td>
<td><code>two_numbers('B21A')</code></td>
</tr>
<tr>
<td><code>two_numbers('hello')</code></td>
<td><code>two_numbers('BB56C')</code></td>
</tr>
</tbody>
</table>
(15) (2 pts.) What will be a result of executing the following Python code:

```python
def best(greeting):
    """(str) -> None
    print a greeting
    """
    start = greeting.find('Best')
    print(greeting[start:])
    return #None

>>> best('CIS122: Best wishes for a pleasant summer.')
Best wishes for a pleasant summer.