20 Points – Do any 20; do more for extra credit. Correct answers are worth +1 point, blank answers are worth 0 points, but wrong answers are incur a −½ point penalty; if you don’t know an answer, leaving it blank is usually better than a bad guess. The following statements have all been executed:

```
Mercury = 5
Venus = 2.5
Mars = [4, 5, ["Phobos", "Deimos"], 7.5, 5, 3]
Jupiter = ["Ganymede", "Callisto", "Io", "Europa"]
Saturn = ("Titan", "Iapetus", "Mimas")
```

Show the computed result for each problem; all are independent of one another. Indicate where a computation fails because of some form of error. Be careful about the type of the result, particularly int, float, bool, and complex types, and put proper quotes around string results, square brackets around lists, and parentheses around tuples.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Mercury / 2</td>
<td>2.5 (float)</td>
</tr>
<tr>
<td>2. Mercury // 2</td>
<td>2 (int)</td>
</tr>
<tr>
<td>3. Mercury % 2</td>
<td>1 (int)</td>
</tr>
<tr>
<td>4. Venus * 2</td>
<td>5.0 (float)</td>
</tr>
<tr>
<td>5. Venus &gt; 1</td>
<td>True (bool)</td>
</tr>
<tr>
<td>6. len(Venus)</td>
<td>ERROR</td>
</tr>
<tr>
<td>7. len(Mars)</td>
<td>6 (int)</td>
</tr>
<tr>
<td>8. len(Mars[2])</td>
<td>2 (int)</td>
</tr>
<tr>
<td>9. len(Mars[2][1])</td>
<td>6 (int)</td>
</tr>
<tr>
<td>10. Mars[1]</td>
<td>5 (int)</td>
</tr>
<tr>
<td>11. Mars[2][1]</td>
<td>&quot;Deimos&quot; (string)</td>
</tr>
<tr>
<td>12. Mars[2][1][3]</td>
<td>&quot;m&quot; (string)</td>
</tr>
<tr>
<td>14. Jupiter[-1]</td>
<td>&quot;Europa&quot; (string)</td>
</tr>
<tr>
<td>15. Jupiter[1:3]</td>
<td>[&quot;Callisto&quot;,&quot;Io&quot;] (list)</td>
</tr>
<tr>
<td>16. Jupiter[1:]</td>
<td>[&quot;Callisto&quot;,&quot;Io&quot;,&quot;Europa&quot;]</td>
</tr>
<tr>
<td>17. Jupiter[:3]</td>
<td>[&quot;Ganymede&quot;,&quot;Callisto&quot;,&quot;Io&quot;]</td>
</tr>
<tr>
<td>18. Jupiter + Saturn</td>
<td>ERROR</td>
</tr>
<tr>
<td>19. list(range(Mercury))</td>
<td>[0,1,2,3,4] (list)</td>
</tr>
<tr>
<td>20. list(range(4,23,Mercury))</td>
<td>[4,9,14,19] (list)</td>
</tr>
<tr>
<td>21. list(range(len(Saturn[1])))</td>
<td>[0,1,2,3,4,5,6] (list)</td>
</tr>
<tr>
<td>22. list(range(Mars[-1],Mars[1]))</td>
<td>[3,4] (list)</td>
</tr>
<tr>
<td>23. [X*Mercury for X in range(3)]</td>
<td>[0,5,10] (list)</td>
</tr>
<tr>
<td>24. [X for X in Saturn[2]]</td>
<td>[&quot;M&quot;,&quot;i&quot;,&quot;m&quot;,&quot;a&quot;,&quot;s&quot;] (list)</td>
</tr>
<tr>
<td>25. [0 for X in range(len(Jupiter))]</td>
<td>[0,0,0,0] (list)</td>
</tr>
</tbody>
</table>
5 Points – Find and correct all the syntax errors in the following function:

```python
define MyFunction (Frog,Toad,Newt=4):  
    Total = 0  
    while Item in [5,2,8,1,9]:  
        Total = Total + Item × (Frog + Toad / Newt)  
    return Total
```

1 point each; give +1 extra credit if they find all six, -½ for each non-error misidentified as an error.

10 Points – Write a counter-loop where the counter variable is called Frog, the initial value is 14, the test value is 39, and the step value is 3. The payload of the loop is to print Frog. Your solution should take at most four lines of Python and must not contain any def or return statements.

```python
Frog = 14
while Frog < 39:
    print (Frog)
    Frog = Frog + 3
```

20 Points (5 points each) – Show what is printed out as the result from calling Main() (four lines total):

```python
def F1(X,Y,Z=2):
    Q = X + Y × Z
    return Q - 1

def F2(Z,X=1,Y=3):
    return F1(Y,Z,X)

def F3(X,Z,Y):
    return F2(F1(X+Y,Z))

def Main():
    print (F1(2,2,3))  # Answer #1: 7
    print (F2(4,3))    # Answer #2: 14
    print (F3(3,2,4))  # Answer #3: 12
    print (F1(3,-2))   # Answer #4: -2
return
```
15 Points – Trace the following flowchart. Any time a variable value is updated, **write the new value** in the appropriate box, **crossing out any previous value**. There will be several lines printed as a result of running the program; show the values of the variables in each of those lines. There will be fewer than eight lines printed. Three points per line; one point for each of the three values. Can be in orders other than N,P,T as long as they are properly labeled; if in an odd order and not labeled remove 1 point overall. Remove 1 point overall if they didn’t do something “reasonable” in the variables box. Remove 2 points per extra line if there are more than five lines of output.
15 Points – Convert the flowchart on the previous page into the equivalent and correct Python 3 code. Your solution won’t need any function definitions, and you will lose points if you use `def` or `return`.

```python
T = 0
P = 1
N = 0
while N < 5: 
    P = P * 2
    T = T + P
    N = N + 1
print (N,P,T)
```

Remove 1 point per error for any of the following problems:
- Statements in a different order from the flowchart
- Syntax error
- Change in statement semantics from flowchart
15 Points – Complete the following function `Bracket` to return `True` if `N` is greater than or equal to -100 and less than or equal to +100, but it must return `False` if `N` is either less than -100 or greater than +100. Your solution must NOT contain any `print` statements. Accept any of the following forms (or any other form that works). Remove 1 point per syntax error or logic error.

```python
def Bracket(N):
    if (N >= -100) and (N <= 100):
        Result = True
    else:
        Result = False
    return Result
```

```python
def Bracket(N):
    if (N < -100):
        Result = False
    else:
        if (N > 100):
            Result = False
        else:
            Result = True
    return Result
```

```python
def Bracket(N):
    if (N < -100):
        Result = False
    elif (N > 100):
        Result = False
    else:
        Result = True
    return Result
```

```python
def Bracket(N):
    if (N < -100): Result = False
    elif (N > 100): Result = False
    else: Result = True
    return Result
```

```python
def Bracket(N):
    Result = True
    if (N < -100): Result = False
    if (N > 100): Result = False
    return Result
```

```python
def Bracket(N):
    return (N >= -100) and (N <= 100)
```